(2)

Guide for the Management of Expert Systems Development

Iris Kameny, Umar Khan, Jody Paul, David Taylor

The research described in this report was sponsored by the Assistant Secretary of Defense for Production and Logistics. The research was conducted in the National Defense Research Institute, RAND's federally funded research and development center supported by the Office of the Secretary of Defense, Contract No. MDA903-85-C-0030.

ISBN: 0-8330-0991-5

Published by The RAND Corporation
1700 Main Street, P.O. Box 2138, Santa Monica, CA 90406-2138

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER R-3766-P&L	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Guide for the Management of Expert Systems Development		5. TYPE OF REPORT & PERIOD COVERED interim 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Iris Kameny, Umar Khan, Jody Paul, David Taylor		8. CONTRACT OR GRANT NUMBER(s) MDA903-85-C-0030
9. PERFORMING ORGANIZATION NAME AND ADDRESS The RAND Corporation 1700 Main Street Santa Monica, CA 90406		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Commander, U.S. Armed Forces Medical Intelligence Center, Fort Detrick Frederick, MD 21701-5000		12. REPORT DATE July 1989 13. NUMBER OF PAGES 198
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

No Restrictions

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Artificial Intelligence
Computer Applications
Computer Programs
Risk
Evolution (Development)
Standards

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

see reverse side

DD FORM 1 JAN 73 1473

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

This report was prepared to assist managers in the risk-managed application of expert systems (ES) technology. Two problems that have frustrated managers and technicians who might otherwise be interested in exploring ES solutions are the seeming lack of a software engineering discipline for creating and maintaining expert systems and the fact that the software development risks associated with expert systems have not been clearly addressed. This guide focuses on ES development as a software engineering endeavor and emphasizes that, as such, it should be subject to much the same discipline as conventional automated information system software. The guide is meant to be used in conjunction with accepted software development methods and standards, such as Department of Defense (DOD) Standards 2167A and 7935.1, DOD Directive 7920.1, DOD Instruction 7920.2, and the OSD Major Automated System Review Council (MAISRC) Guidelines for Program Managers. This document presents a step-by-step guideline that addresses the managerial and technical risks of each phase in the software development life cycle of systems that contain ES components. (See also N-2970.)

R-3766-P&L

Guide for the Management of Expert Systems Development

Iris Kameny, Umar Khan, Jody Paul, David Taylor

July 1989

Prepared for the
Assistant Secretary of Defense
(Production and Logistics)

RAND

PREFACE

This report is a step-by-step Guide for the management of expert systems development, prepared to assist managers in the risk-managed application of expert systems technology. It addresses the managerial and technical tasks of each phase in the software development life cycle of systems that contain expert system components. *Guide for the Management of Expert Systems Development: Additional Appendixes* is a RAND Note (N-2970-P&L, August 1989) by the same authors that contains Appendixes C and D to this report.

This research was sponsored by the Assistant Secretary of Defense (Production and Logistics). It was conducted under the Information Processing Systems Program within The RAND Corporation's National Defense Research Institute, a federally funded research and development center sponsored by the Office of the Secretary of Defense.

The Office of the Secretary of Defense is preparing a concise *Quick Reference Guide* to accompany this important reference. This companion Guide will emphasize the differences between traditional software development and expert systems development for program managers and other developers. Using detailed references to the RAND Guide, the *Quick Reference Guide* will make it easy to find, understand, and use the wealth of knowledge incorporated in the RAND document. This Guide will be available early in 1990 from the Office of the Assistant Secretary of Defense (Production and Logistics), Directorate for Automation Support and Technology.

SUMMARY

This report has been prepared to assist managers in the Department of Defense (DoD) in determining the applicability of expert system (ES) technology as an automation solution. It is also intended to serve as a guide for production and logistics (P&L) managers and technicians seeking ES solutions to specific problems. This Guide should be a living document; hopefully, its use in the practical solution of P&L problems using ES technology will generate feedback from the responsible managers and technicians, thus leading to future versions which will continue to be responsive to the P&L community's needs.

Two problems have frustrated managers and technicians who might otherwise be interested in exploring ES solutions: (1) the seeming lack of a software engineering discipline for creating and maintaining expert systems and (2) the fact that software development risks associated with expert systems have not been clearly addressed. The position taken here is that ES development is a software engineering endeavor and, as such, should be subject to much the same discipline as conventional automated information system (AIS) software. This Guide is meant to be used in conjunction with accepted software development methods and standards, such as DoD Standards 2167A [1987] and 7935.1 [1984], DoD Directive 7920.1 [1978], DoD Instruction 7920.2 [1978], and the DoD/OSD "Major Automated Information System Review Council (MAISRC) Guidelines for Program Managers" [MAISRC 1987].

ORGANIZATION OF THIS GUIDE

Chapter 1: introduces the Guide, its purpose, objectives, scope, and audience.

Chapter 2: offers an overview of the application of software life-cycle process models to expert system development.

Chapter 3: describes a mapping of the ES life-cycle process to the DoD software documentation and life-cycle process.

Each of the next seven chapters addresses a stage in the development of an ES mapped to the development phases described in DoD Directive 7920.1.

Chapter 4: describes the Initiation Phase, that is, how one determines what the problem is, whether it is feasible to apply ES technology to the problem and, if so, how to begin exploring the problem.

Chapter 5: describes the Concept Phase, that is, how to further explore the problem to decide how to solve it. In-house ES and other support tools are used to develop rapid prototypes of parts of the problem solution.

Chapter 6: describes the Definition/Design Phase, Demonstration Prototype, that is, the rapid design and prototyping of the problem solution by the Knowledge Engineer through interaction with experts and end-users in a stand-alone mode with stubbed-in interfaces to AISs.

Chapter 7: describes the Definition/Design Phase, Testbed Prototype, that is, the redesign and re-implementation of the functionality defined and demonstrated by the Demonstration Prototype using the development tool.

Chapter 8: describes the Development Phase, Operational Prototype, that is, the production of the Operational Prototype, which integrates the Testbed Prototype into an example operational environment using the actual deployed tool, AIS(s), and special devices.

Chapter 9: describes the Deployment Phase, Operational System, that is, delivery of the fully integrated system to the field.

Chapter 10: describes the Post-Deployment Phase, that is, helping maintain the system, evaluating its use after 6 months to determine lessons learned, and planning enhancements to be implemented in system upgrades.

Finally, two appendixes include a tutorial and glossary, and document descriptions. Two further appendixes, documented in a separate RAND publication, N-2970-P&L, address tool evaluation and selection, and case studies. A bibliography is listed at the end of this report.

CONTENTS

FIGURES

TABLES

1. INTRODUCTION

1.1 MOTIVATION

Over the past decade, development and use of expert system (ES) technology in the commercial sector have flourished. In search of better methods for attacking problems, the Department of Defense (DoD) Production and Logistics (P&L) community has recently begun applying ES technology. Most of the early applications of ES technology led to simple stand-alone systems requiring no integration with existing automated information systems (AISs). Since ES technology has proven to be a viable technology when applied to these smaller P&L applications, the next step is to use it in larger applications and especially for problems requiring integration with large software systems such as AISs.

Although the commercial sector has stressed the uniqueness of ES development (using rapid prototyping with extensive user participation) when compared with traditional software development, it has supplied little guidance for managing this unique process. This apparent lack of a software engineering discipline for creating and maintaining expert systems is a challenge that has frustrated both P&L managers and technicians who might otherwise be interested in exploring ES solutions to larger problems. This Guide was motivated by the need to show how existing DoD software development methods and standards (such as DoD Standards 2167A and 7935.1, DoD Directive 7920.1, and the OSD "Major Automated Information System Review Council (MAISRC) Guidelines for Program Managers" [MAISRC 1987]) can be applied to the management of ES development efforts. A recurring theme throughout is that ES development is a software engineering endeavor and is subject to much the same discipline as the development of conventional software systems such as AISs.

1.2 PURPOSE

The purpose of this Guide is to help managers and technicians in the P&L community determine the applicability of ES technology for solving their problems and to guide them in all phases of ES development from initial conception through post deployment.

It provides guidance to managers in assessing the risks of ES development and in the generation of a realistic management plan. At each phase in the life cycle, it describes requirements for staffing and user participation, ES tools and environment, training, quality assurance, test and evaluation, configuration management, security, maintenance, and documentation. It discusses preparation for in-process and milestone reviews and the need to solicit in-house support for the ES effort.

The Guide instructs the technical staff in iteratively defining the problem solutions. It stresses the need for a well-defined technical plan at each life-cycle phase that meets user mission requirements as well as cost and schedule constraints. It stresses the need for good software development techniques, early definition of test cases and test and evaluation activities, and constant monitoring of application changes that may affect integration solutions.

Three major themes are stressed throughout: (1) the importance of and possible difficulties associated with integration, (2) the need for continuous user and domain expert participation throughout the life cycle, and (3) the need for dynamic documentation of the design and development process.

1.3 OBJECTIVES

The main goal of this Guide is to indicate how current DoD software life-cycle development standards can be applied to the management of expert system development. Some additional goals that focus on key managerial and technical issues include:

- Identifying factors related to success or failure in applying ES technology to a problem (this includes understanding the strengths and weaknesses of the technology and the associated trade-offs with more traditional techniques);
- Identifying unique characteristics in the development and operation of an ES that may require treatment different from that used in conventional software development or project management;
- Identifying potential risks in the integration of an ES within larger AIS architectures;
- Identifying acquisition considerations for ES tools and products, training, and maintenance.

1.4 SCOPE

The scope of this Guide is defined in terms of the audience being addressed and the most appropriate types of applications.

The Audience. This Guide is intended for software engineers, developers, and managers who are investigating the potential of ES technology within their organization or customer community and who may be beginning their first ES development effort. It is assumed that readers will have some familiarity with ES technology and its terminology before using this Guide. More important, it is assumed that readers are knowledgeable about DoD software development methods and standards and the management of conventional software development projects.

The Applications. This Guide addresses medium- to large-scale ES development projects—particularly those efforts that require integration with AIS(s). The reference to size is not intended to imply the use of any metric such as numbers of rules or frames. Rather, size is a subjective estimate of complexity, depth of reasoning, and degree of integration with the organization's existing computing base.

This Guide addresses:

- ESs that are deployed on conventional computers, particularly mainframes and workstations, rather than on only specialized platforms such as LISP machines;
- ESs that focus on *applying* the technology rather than *developing* it;
- ESs that attempt to capture knowledge of skilled people (with five to ten years' experience) as well as those systems that embody the knowledge of senior experts (with ten to twenty years' experience);
- ESs that integrate with existing, conventional software—especially database management systems (DBMSs), and, to a lesser degree, spreadsheets, simulation programs, decision support systems (DSSs), other expert systems, and traditional programs; and,
- ESs that are prototyped on one type of hardware using a strong prototyping tool and deployed on a different platform or using a conventional programming language.

Small ES Development Projects. Over the past few years there has been a focus on using ES tools to build small, well-defined, stand-alone applications. This is a logical extension of the trend in the microcomputer industry to put greater development capabilities in the hands of the end-user. Some applications can and should be implemented by end-users using today's ES tools as very-high-level languages, such as is possible with

spreadsheets and DBMSs. But proliferation of copies of such applications leads to management concerns about what constitutes a proper use of this technology. An organization needs to assess its policies as to how and under what circumstances these tools should be used and how ES copies will be controlled and monitored to ensure appropriate use of the technology. This Guide may be useful to developers and managers of these smaller projects, particularly with respect to topics not addressed in the literature such as: using project Progress Notebooks; structuring the prototyping phase; risk management; testing, validating, and evaluating performance; early consideration of deployment needs; and long-term maintenance considerations.

1.5 A MAP TO THE GUIDE

Chapter 1: introduces the Guide, its motivation, purpose, objectives, and scope.

Chapter 2: contrasts the conventional software engineering model, the waterfall model, with the newer spiral model, which is more amenable to ES development. Expert system technology is discussed, some commonly used concepts defined, and differences between conventional and ES software development are noted.

Chapter 3: maps ES life-cycle development into existing DoD software life-cycle development methodology and documentation standards. It discusses three main themes important throughout the ES development process and offers detailed guidance about how to use the rest of the Guide.

Chapters 4–11: address each phase in the development of an ES.

Finally, two appendixes are included: a tutorial and glossary, and a description of DoD-required documents.

2. APPLICATION OF SOFTWARE LIFE-CYCLE PROCESS MODELS TO EXPERT SYSTEM DEVELOPMENT

The goal of a software life-cycle process model is to maximize the probability of developing viable, sustainable software within cost and on schedule. The main functions of the process model are to determine the order of the stages involved in software development and to establish transition criteria for progressing from one stage to another. A process model addresses the questions of: (1) what process should be done next, and (2) what is needed to complete that process. When institutionalized, a software life-cycle process model becomes a "reference model" that can be shared and communicated among the buyer of the software product, the manager of the development effort, the technical staff, the users, and the maintainers. It is the lack of such a model that makes managing the development of expert systems so difficult for P&L managers today.

This chapter begins by describing (1) the waterfall life-cycle process model, which is the basis of current DoD software acquisition standards, and (2) the recently proposed risk-driven spiral life-cycle model developed from refinements to the waterfall model. A discussion of expert system technology follows, emphasizing distinctive ES characteristics. The last sections describe the expert system life-cycle development process used in this Guide and how it relates to the spiral life-cycle model.

2.1 THE WATERFALL MODEL

The "waterfall" model, shown in Figure 2.1, is the basis of current DoD software acquisition standards. It was described by W. W. Royce [1970] and has undergone many enhancements over the years. As described in Boehm [1981, 1988], the waterfall model begins with a system feasibility phase in which a software product concept and life cycle are developed.

The product concept feeds into the software plans and requirements phase, which may feed back to refine the product concept, until the requirements specification of functions, interfaces, and performance is validated against the product concept.

The software plans and requirements feed into the product design phase, which may feed back into the plans and requirements phase, until there is a complete product specification of the hardware/software architecture, control structure, and data structure that is valid against the requirements specification.

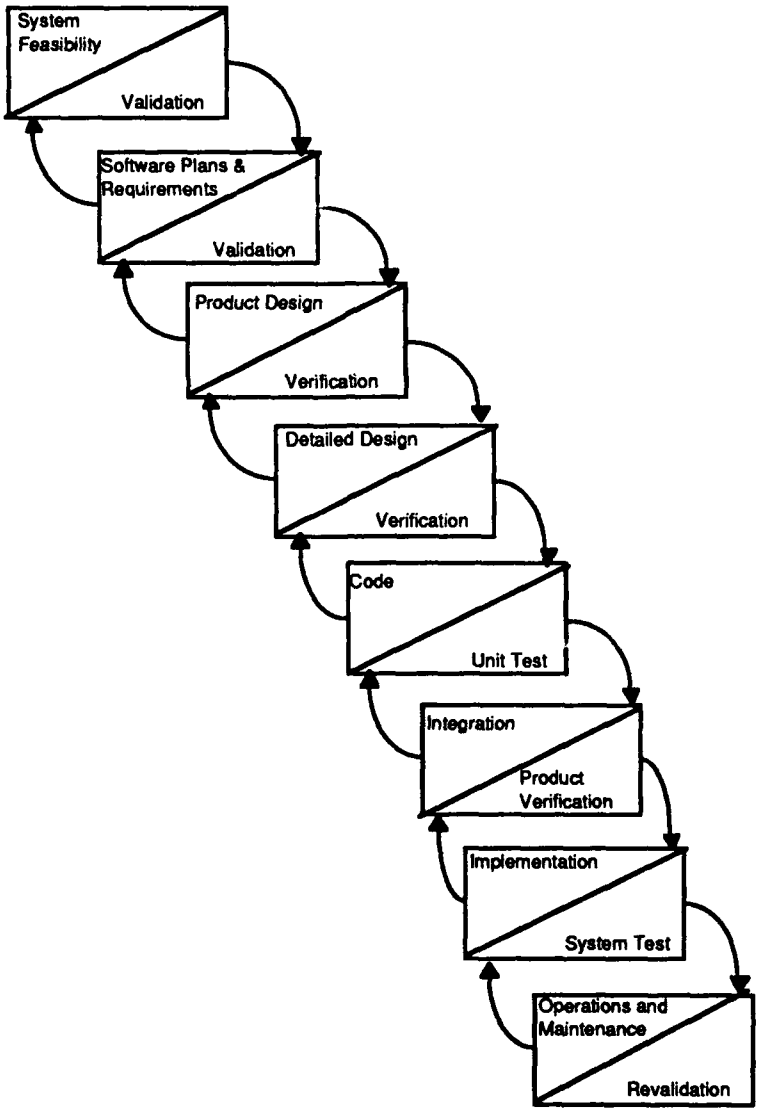
The product design specification feeds into the detailed design phase, which may feed back into the product design phase, until there is a complete, verified specification of the control structure, data structure, interface relations, sizing, key algorithms, and assumptions of each program component.

The detailed design feeds into the coding phase, which may feed back into detailed design, until agreement and validation of code with design are assured. The code phase output is a complete set of program components that have been verified by unit testing.

The unit tested program components feed into the integration phase where they are integrated into a properly functioning product. Integration feeds back into the coding phase and may result in component coding changes to accommodate unforeseen integration problems.

The product feeds into the implementation phase during which any program and data conversion needs are met, as well as installation and training. The installed system then undergoes system test.

Finally, during the operations and maintenance phase, each iteration to update the product is revalidated through system test.



SOURCE: Boehm [1988], © 1988 IEEE.

Figure 2.1—Waterfall software development model

The waterfall model's approach was an improvement over earlier process models because it incorporated feedback loops between successive stages and used the initial incorporation of prototyping via a "build it twice" step running in parallel with requirements analysis and design.

Many refinements have been added to the waterfall model over the years, but none has addressed the major sources of difficulty: restrictive interpretation of the life-cycle phases and the emphasis on fully elaborated documents as completion criteria for early requirements and design phases. Although the waterfall model may work for well-defined and understood applications such as compilers and payroll packages, it does not work well for some classes of software, particularly interactive end-user applications for which the user interfaces and decision support functions may be poorly understood [Boehm 1988]. In these types of applications, the inflexible ordering of the software phases—particularly postponing implementation and user feedback until the design is complete—may put the project at great risk. Document-driven standards may result in the design and implementation of large amounts of code discovered to be unusable during the integration or implementation phases.

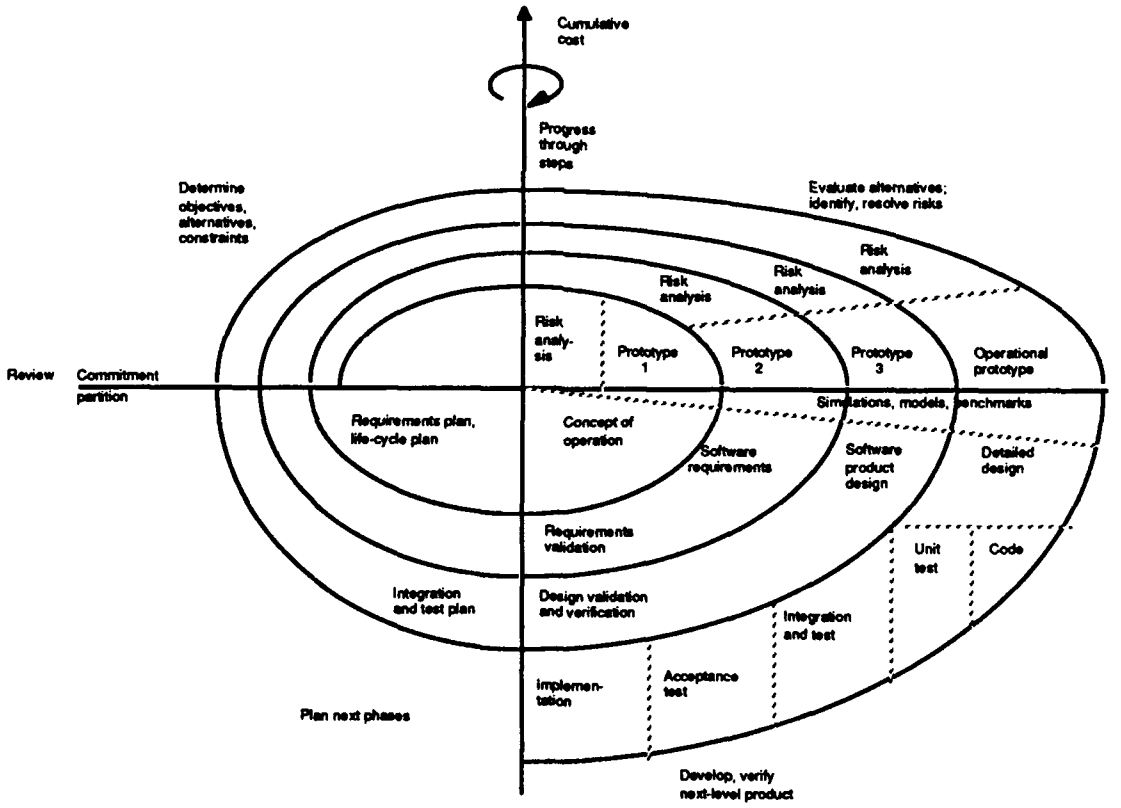
"Build it twice" does not necessarily answer the problem if the best solution is incremental development of a prototype into the final solution.

Boehm has suggested refining the waterfall model to allow for incremental development [Boehm 1981]. This would allow developers to develop, test, and validate software in increments of functional capability, making it easier to test and allowing earlier validation of the product by the users. This and other suggested refinements to the waterfall model have led to development of the spiral model, which is discussed in the next section.

2.2 THE SPIRAL MODEL

The spiral model of the software process has evolved out of experience in using the waterfall model for large government software projects. It was developed to accommodate most previous process models as special cases and to provide guidance for modeling a given software situation [Boehm 1988].

In the spiral model, Figure 2.2, the radial dimension represents cumulative cost and the angular dimension represents progress made in completing each cycle of the spiral. Each cycle of the spiral proceeds through the steps shown beginning in the upper left quadrant. Each new cycle begins by defining the objectives of the software product in terms of characteristics such as performance, functionality, and flexibility. Alternative implementations are identified along with the constraints imposed by each alternative (e.g., cost, schedule, interface).



SOURCE: Boehm [1988], © 1988 IEEE.

Figure 2.2—Spiral software development model

Boehm [1988, page 65] describes the next group of steps:

> The next step is to evaluate the alternatives relative to the objectives and constraints. Frequently, this process will identify areas of uncertainty that are significant sources of project risk. If so, the next step should involve the formulation of a cost-effective strategy for resolving the sources of risks. This may involve prototyping, simulation, benchmarking, reference checking, administering user questionnaires, analytic modeling, or combinations of these and other risk-resolution techniques.
>
> Once the risks are evaluated, the next step is determined by the relative remaining risks. If performance or user-interface risks strongly dominate program development, or internal interface-control risks, the next step may be an evolutionary development one: a minimal effort to specify the overall nature of the product, a plan for the next level of prototyping, and the development of a more detailed prototype to continue to resolve the major risk issues.
>
> If this prototype is operationally useful and robust enough to serve as a low-risk base for further product evolution, the subsequent risk-driven steps would be the evolving series of evolutionary prototypes going toward the right in [Figure 2.2]. In this case the option of writing specifications would be addressed but not exercised. Thus, risk considerations can lead to a project implementing only a subset of all the potential steps of a model.

Finally, the last quadrant prepares the plans for the succeeding cycle. During this step the product could be partitioned into several components, each to be developed in a separate parallel spiral cycle.

"The primary advantage of the spiral model is that its range of options accommodates the good features of existing software-process models, while its risk-driven approach avoids many of their difficulties" [Boehm 1988, page 69]. The current difficulties in using the spiral model are: applying it to contract software acquisition, which requires contracts with well-specified deliverables; the need for software developers and managers with expertise in risk assessment; and the lack of adequate elaboration of the process steps which is necessary to ensure that all software development participants are operating in a consistent context [Boehm 1988].

The spiral model is a risk-driven approach to software engineering. Boehm suggests that one characteristic technique from this approach that can be adapted to any life-cycle model is the use of a risk management plan to ensure "that each project makes an early identification of its top risk items, develops a strategy for resolving the risk items, identifies and sets down an agenda to resolve new risk items as they surface, and highlights progress versus plans in monthly reviews" [Boehm 1988, page 71].

In the next section we discuss expert systems and the characteristics that make them different from conventional software applications. The last section describes our approach to managing the development of expert systems—modeled very heavily on the prototyping sector of the spiral model (Figure 2.2) utilizing risk management techniques.

2.3 EXPERT SYSTEM TECHNOLOGY

Waterman defines an expert system as "a computer program that uses expert knowledge to attain high levels of performance in a narrow problem area. These programs typically represent knowledge symbolically, examine and explain their reasoning processes, and address problem areas that require years of special training and education for humans to master" [Waterman 1986, page 390].

2.3.1 Terminology

There are a great number of terms associated with expert systems technology. There is also a widespread lack of standardization within the industry as to the precise meaning and use of many of these terms. For this reason, a Glossary has been provided in Appendix A

giving the definitions used within this Guide. However, there are a few concepts that we will clarify here since their understanding is fundamental to the use of this Guide.

Expert Systems and Knowledge-Based Systems. Throughout this Guide, the popular term *expert system* (ES) is used in preference to the more accurate term *knowledge-based system* (KBS). Figure 2.3 shows the relationship among expert systems, knowledge-based systems, and the general class of artificial intelligence (AI) programs. The figure indicates that most expert systems are built using the knowledge-based approach. In common parlance, the terms *expert system* and *knowledge-based system* are used interchangeably to refer to the general class of systems applying knowledge-based techniques.

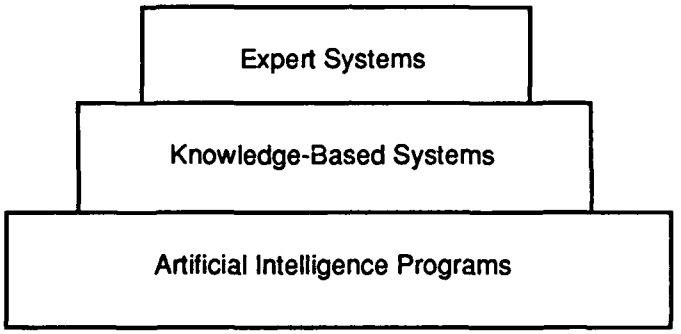


Figure 2.3—Knowledge-based and expert systems

The distinction between these two terms can be understood in the context of how humans perform tasks, such as those to be automated using this technology. Humans perform tasks with varying levels of expertise (e.g., novice, apprentice, journeyman, expert). So, too, do computer systems. In the expert systems setting, the term *expert* implies the automation of the same narrow specialization and high level of competence that define a human expert. An expert system is an automated system that performs at an expert level. The term *knowledge-based system* is intended to focus attention on the knowledge the system contains rather than on whether such knowledge constitutes expertise. Although many of the famous early applications of this technology were, indeed, expert systems, much current work is being done using this technology to provide automated assistants rather than full-fledged experts. Use of the term knowledge-based system avoids confusion over the imprecise and academic boundary of what qualifies as an expert.

Figure 2.4 illustrates the relationship among knowledge, information, and data. Automated information systems (AISs) deal with information to the level of aggregated data. Expert systems work with higher levels of information as well, including both knowledge and meta-knowledge. Visualizing this relationship may help readers to understand the distinction between expert systems and information systems.

Expert System Building Tools and Shells. Expert system building tools are software environments used for constructing expert systems. They include very-high-level programming (knowledge engineering) languages and extensive support facilities. These *knowledge engineering languages*, along with an assortment of support facilities such as debugging aids, explanation facilities, run-time data-acquisition facilities, and knowledge-base editors, are the key AI tools for ES development. *Knowledge engineering environments* are often referred to as *expert system shells.*

2.3.2 Distinctive Expert System Characteristics

We now examine major characteristics of ESs that distinguish them from traditional software applications. They are: knowledge representation and program control strategies, the exploitation of specialized man-machine interfaces (such as built-in explanation facilities), and the impact of commercial ES building tools on development.

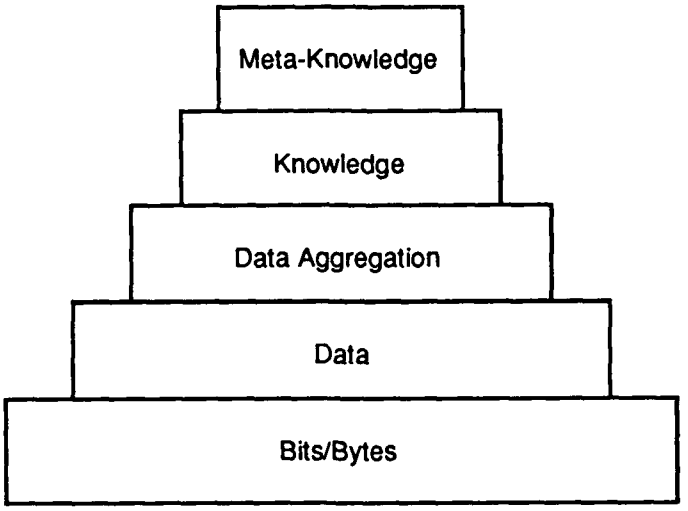


Figure 2.4—Classes of information

Knowledge Representation and Control. Knowledge representation is concerned with "writing down descriptions of the world in such a way that an intelligent machine can come to new conclusions ... by formally manipulating those descriptions" [Brachman 1985]. The decoupling of an expert system's heuristic knowledge from program control is perhaps the most significant contribution of ES technology to traditional software engineering methods.

In expert systems, knowledge that was previously embodied in programming language statements is split: domain and task-specific knowledge are separated from general-purpose knowledge and inference (control) mechanisms. In a conventional program, such as one written in FORTRAN or COBOL, program control (e.g., DO loops, CASE statements, and IF...THEN...ELSE statements) is interwoven with the knowledge (e.g., the algorithm). In an ES, knowledge (e.g., the heuristics) is separated from program control (e.g., the inference engine). The inference engine understands how to manipulate knowledge-base components in general. This allows programming in a data-driven manner, as opposed to the strictly process-driven style of more conventional software.

Several useful representation paradigms have evolved among ES developers. Appendix A includes a discussion of the most common, including: rule-based, semantic nets, frame-based, object-oriented, procedure-oriented, predicate logic based, induction, and access-oriented. The paradigms are supported by individual commercial tools to varying degrees. Traditional software engineering documentation and methodology requirements must be adapted to accommodate ES programming and knowledge representation strategies.

Specialized Man-Machine Interfaces. The heuristic nature of expert system technology and the diverse requirements of ES users drive a need for specialized man-machine interfaces not generally necessary in other types of software. The most significant aspect is enhanced explanation ability. *Explanation* refers to the process whereby the system reveals its knowledge and behavior to the user. The increased complexity of expert systems escalates the need to understand what the system does, how and why it does it, and how to interact with it. A conventional software system is not responsible for describing its behavior to users nor is it expected to convince them of its validity. For example, a word processing system operator is reasonably certain that the program works as expected and does not require justification of how internal decisions are made. On the other hand, the user of a system that recommends therapy for intensive-care patients needs to ascertain the credibility of the suggestion as well as the reasoning involved.

The tasks addressed by expert systems require heuristic knowledge and expertise because simple or algorithmic approaches are not available. Each case processed may result

in a complicated and unique chain of reasoning. Since there is no single algorithm embodied in the software, the reasoning for each situation needs to be explicated and justified. In contrast, a conventional software system, such as one that performs accounting functions, will perform the same way each time. The algorithm it uses can be scrutinized and the implementation code examined to assure correct behavior and accuracy.

Expert system explanation that reveals system knowledge and operation also eases development and maintenance by providing assistance in testing, debugging, and refinement. The system's logic and beliefs need to be examined and their correctness verified if errors appear or uncertainty arises. A domain expert who has improved comprehension of what an existing system does can also grasp its limitations. This provides the basis for designing natural improvements to the system. Such comprehension is prerequisite for formalizing and codifying additional domain knowledge as well.

Impact of Commercial Tools on Development. Expert system development differs from conventional software engineering in its use of specialized tools or shells that provide a rich development environment and a general, very-high-level knowledge engineering language for application programming. Similar trends in software engineering are the development of application-specific fourth-generation languages (4GLs) and CASE (computer-aided software engineering) tools.

Today's ES tools affect the development of expert systems in two ways: the selection process must emphasize functionality contributed by the tool to the ES; and developers now have the option to use different tools for different phases of the life cycle.

Commercial knowledge-based-system building tools differ from each other in how they represent knowledge in the knowledge base (e.g., rules, frames); the inference strategies they support (e.g., forward chaining, backward chaining); the extent and type of interfaces they provide for the developer (e.g., program libraries, reusable source code) and the end-user (e.g., windows, graphics); and their support for integration with external hardware and software (e.g., specific application layer protocols). The better commercial tools help programmers concentrate more of their energies on design problems and less on implementation details.

To paraphrase Peter Szolovitz [1987], the principal objective of evaluating tool systems is to determine their fit to the problems under consideration. As in the case of evaluating third-generation languages, there is probably no "best" language overall, only a variety of well and poorly designed tools that make different assumptions about the sorts of problems to which they will be applied and that make different design trade-offs which affect the cost and performance of those tools. Later chapters and Appendix C in the supplementary Note [Kameny 1989] discuss the technical process of tool selection.

2.4 THE EXPERT SYSTEM LIFE-CYCLE PROCESS MODEL

Figure 2.5 shows the six phases of the expert system life-cycle process supported by this Guide: initiation, concept, definition/design, development, deployment, and post deployment. As will be seen in Chapter 3, these phases can be mapped into the DoD software life-cycle process using an approach more closely aligned with the risk-driven spiral model than with the waterfall model currently supported by DoD standards. The ES life-cycle process involves several phases of prototyping similar to the sector of advancing prototypes shown in the Figure 2.2 spiral software development model.

The main characteristic of our approach to ES development is the reduction of risk by developing an ever increasing understanding of the problem, user needs, and requirements through demonstration and use. Major application issues such as user interface, integration, and security are addressed in every phase. All stages of development emphasize the need for a team support effort of ES developers, users and experts, maintenance people, security and systems people, and management.

The ES life-cycle model requires four stages of prototyping. The output of each stage is a product that demonstrates lessened risk and better problem understanding and is closer to the final problem solution. The product of the final prototyping stage is a system ready for deployment. Each prototyping phase may be composed of several iterations, each planned to add a well-defined increment to the problem solution.

The first prototyping phase, the Concept Prototypes, concentrates on exploring potential high-risk areas of the problem with experts and users to determine if they can be feasibly dealt with. Next, the Demonstration Prototype focuses on problem definition by iteratively exploring, scoping, and defining the problem space. This results in a stand-alone ES with stubbed-in input/output connections to external systems in lieu of real connections. This is followed by the Testbed Prototype, which focuses on the development of a verified and validated redesign and re-implementation of the Demonstration Prototype on the deployment platform. Finally, the Operational Prototype, which will be the baseline system, is the integrated system focusing on performance and system acceptance testing.

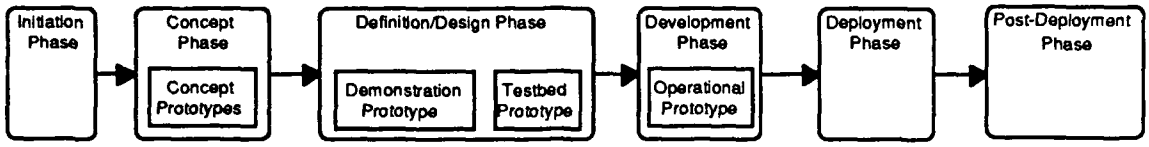


Figure 2.5—Expert system development life cycle

The *Initiation Phase* is concerned with deciding what is wanted. This phase deals with determining what the problem is, whether ES technology can be feasibly applied to the problem, how to begin exploring the problem, and whether the problem solution satisfies the users' mission needs. The culmination of this phase is a Milestone 0 review which, if successful, brings the project to the next phase.

The *Concept Phase* involves further exploration of the problem and deciding how best to solve it. In-house ES building tools and other support tools are used to develop rapid prototypes of parts of the problem solution. These *Concept Prototypes* focus on those items identified as high risk. The result of this activity is not a single integrated concept prototype but rather many small prototypes. These may demonstrate: how the general problem solution will work, how parts of the problem solution that strain ES capabilities might be handled, how the user interface may be structured, how integration with AIS(s) or special devices will be handled, and how security may be handled.

The *Definition/Design Phase* is concerned with the development of the final design for the problem solution. This design is embodied in a working prototype and associated documentation. The phase is divided into two stages, corresponding to the two prototypes used for design development: the Demonstration Prototype and the Testbed Prototype. Each of these stages culminates in a Milestone 2 review.[1]

The *Demonstration Prototype* focuses on rapid design and prototyping of the problem solution in a stand-alone mode by the Knowledge Engineer (KE) through extensive interaction with the expert(s) and to a lesser degree with end-users. The ES tool used in this stage is primarily selected for its ability to support the design effort and should offer extensive support for debugging, graphics, knowledge acquisition, multiple modes of reasoning, and knowledge representation. The major goal of this stage is to correctly demonstrate the problem solution with an adequate user interface. The Demonstration Prototype is not directly concerned with fast execution (tool choice may be the result of selecting development functionality and speed over fast operational execution). The interfaces with AIS(s) and special devices should at least be stubbed and the data exchanges simulated. Security may be designed and stubbed, but the underlying support (e.g., from the

[1] Another effect of risk management is the ability to recognize when the problem solution is not feasible and the effort should be halted. Even in this situation, there is a partial product—the region of the problem domain that has been explored and documented.

operating system) may be lacking in the Demonstration Prototype software base. As a minimum, a security proof-of-concept design should be demonstrated by a walk through.

The *Testbed Prototype* is a redesign and re-implementation of the functionality defined and demonstrated in the Demonstration Prototype, possibly plus enhancements. It may be implemented using the ES tool (or the developer's version of the tool) that will be used in the deployed system or it may be coded directly in a DoD-acceptable programming language (ES tools make it possible to write expert systems more complex than could be practically implemented in a conventional language). In the case where the Demonstration Prototype and the Testbed Prototype use the same tool or family of tools, we still strongly recommend re-implementation of the Testbed Prototype rather than incrementally evolving it from the Demonstration Prototype. This is more likely to ensure a cleaner design and implementation that will be easier to maintain through the rest of the development phases.

Correctness is required of the Testbed Prototype, and formal test and evaluation (T&E) procedures should validate that it meets its requirements as specified in the Functional Description (FD). Fast operational execution may be a requirement at this stage or may be delayed until the Operational Prototype is developed if different versions of the tool are used in the Testbed and Operational Prototypes. Again, integration interfaces will be implemented but will not be directly linked into an AIS. Security aspects of the workstation environment should be implemented and tested.

The *Development Phase* produces the *Operational Prototype,* which is the result of integrating the Testbed Prototype into an example operational environment using the actual deployed tool, AIS(s) and special devices. The ES tool may differ between the Testbed and Operational Prototypes if a version of the tool with development support is used in the Testbed and a special deployment version (in which development support has been removed to reduce size and enhance performance) is used for the Operational Prototype. Correctness and performance should be validated as should requirements for integration and security. At the completion of the Operational Prototype, the ES is ready for deployment.

The final Operational Prototype is the *baseline system*. However, in order to maintain the ES it may be necessary to iterate back to various prototype stages. For example, small bugs can be corrected or modifications properly made to the ES baseline but larger changes in the knowledge base or knowledge-based processes will require ES tool development support; they may also have to be implemented and tested as part of the Testbed Prototype, then ported to the Operational Prototype for further testing, and finally released and re-installed at deployed sites. Major upgrades may require building Concept Prototypes or using the existing Testbed Prototype as the basis for developing a new Demonstration Prototype and then reiterating through the various phases of development. For a mature ES, there could be different phases of upgrades under way concurrently, each at a different stage of development.

The Development Phase culminates in the Milestone 3 review which, if successful, carries the project to the Deployment Phase.

In the *Deployment Phase,* the fully integrated system is delivered to the field. This phase culminates in a Milestone 4 review which, if successful, results in the system being turned over to the support organization that will maintain it.

The *Post-Deployment Phase* is concerned with maintenance of the system, evaluation of the use of the system to determine lessons learned, and planning of enhancements to be implemented in system upgrades.

2.5 USE OF THE RISK-DRIVEN SPIRAL MODEL FOR ES DEVELOPMENT

Risk-Driven Approach. This Guide advocates use of a risk-driven approach to expert system development to enhance the likelihood of success or early recognition of infeasibility within cost and schedule constraints. This is an appropriate approach for a new technology that addresses problems that are often ill-defined and require heuristic solutions. In risk-

driven ES development, resources are focused on exploring solutions to high-risk areas before committing to lower-risk activities, thus minimizing costs in case the project is terminated due to limitations in funding, time, staffing, or technology. The cost estimate for an expert system development effort, and thus the return on investment (ROI), begins as a ballpark estimate and becomes increasingly better substantiated as the prototyping proceeds and the problem and its solution become better defined.

Use of Spiral Model. The heuristic nature of ES applications and the intensive expert and user interactions required to develop the knowledge base and, in many cases, the user/machine interactions make these applications similar to those Boehm identifies as poorly served by the waterfall model and better served by the spiral model (e.g., software applications in which user interfaces and decision support functions are poorly understood) [Boehm 1981, 1988]. The ES development technique entailing the implementation of ever advancing, more-functionally-complete prototypes is similar to the "sector" of prototypes supported by the spiral model. The ES development process, driven by risk-management, is an ever-spiraling mixture of requirements definition, design, and development that does not map well into the explicit process stages of the waterfall model.

Difficulties with Using the Spiral Model. Let us now address the difficulties Boehm noted in using the risk-driven spiral model: (1) it does not address the well-specified deliverables required for contract software acquisition; (2) it requires software developers and managers to have expertise in risk assessment; and (3) it does not adequately elaborate the process development steps to ensure that all software development participants operate in a consistent context [Boehm 1988].

This Guide addresses the first difficulty by recognizing the unreliability of preliminary costing estimates and by guiding managers (1) to seek incremental funding for ES development efforts; (2) to develop prototypes (particularly Demonstration and Testbed Prototypes) as a set of iterations in which the objectives for each iteration are well-defined and costed; (3) to scope the objectives of each iteration to fit cost and time schedules while being constrained to still meet mission objectives; (4) to acquire and apply cost information from early prototype iterations to estimates of later cycles; and (5) to present better substantiated cost estimates at each milestone review.

Regarding the second difficulty, the Guide does not offer training in risk assessment for those managers lacking the expertise, but it does point out the authors' views of the ten most likely "generic" risks at each development stage. These could be used by a manager as a guide in developing his/her own ten highest risks based on the characteristics of the application.

The main purpose of this Guide is to address the third difficulty—we offer detailed elaboration of the process development and information management steps required for expert system development for both managers and technicians.

In the next chapter, we show how the expert system life-cycle process can be mapped into the DoD life-cycle development process, standards, and documentation.

3. MAPPING THE ES LIFE-CYCLE PROCESS TO THE DoD SOFTWARE DOCUMENTATION AND LIFE-CYCLE PROCESS

Expert system life-cycle development, like conventional software life-cycle development, requires management controls to ensure that all requirements are satisfied and that proper testing of each development stage is planned and executed. Over the past four or five years, the market has been dominated by platform-specific commercial shells aimed at the development of stand-alone systems. This has provided ES developers with greater experience in the first four phases than in the Deployment and Post-Deployment Phases. The main objective of this chapter is to show how expert system development can be managed from initiation through post deployment using existing DoD standards.

This chapter begins by showing how the expert system development process described in Chapter 2 can be mapped to DoD Directive 7920.1 [June 20, 1988]. Documentation requirements are then discussed in relation to draft DoD Standard 7935.1 [January 15, 1988] noting exceptions for expert system development and introducing a new document concept, the Progress Notebook. Three major development themes are discussed: integration concerns, need for continuous user and domain expert participation, and need for dynamic documentation. Finally, we describe the way in which the remaining chapters are organized.

3.1 MAPPING EXPERT SYSTEM DEVELOPMENT TO DoD DIRECTIVE 7920.1

DoD standardizes and regulates the sequencing of stages for software development in order to effectively manage the transformation of end-user requirements into a proposed automated system. DoD Directive 7920.1 [June 20, 1988] addresses life-cycle management of large automated information systems having anticipated program costs in excess of $100 million or in excess of $25 million in any single year. Although such systems are larger than we would expect an expert system to be, expert system life-cycle phases can be mapped straightforwardly into the AIS life-cycle management phases of 7920.1 as shown in Figure 3.1 and discussed below.

Needs Justification Phase. The purpose of the AIS Needs Justification Phase is to define and document a mission need and validate that need in a Mission Need Statement (MNS). It maps to the ES Initiation Phase, which is concerned with understanding the problem, determining the feasibility of an ES solution, and assessing the mission need with respect to the scope of the problem solution. Each phase ends with a Milestone 0 review which, if successful, indicates that the mission need is validated and it is feasible to continue the effort.

Concepts Development Phase. The purpose of the AIS Concepts Development Phase is to identify and evaluate alternative functional and technical concepts that satisfy the MNS and select the best program to implement the required capabilities. It maps to the ES Concept Prototypes Phase, which concentrates on exploring alternative solutions to potential high-risk areas of the problem in order to demonstrate feasibility. Both the AIS and ES efforts address evaluation and selection of development and acquisition strategies, and initial planning for design, development, testing, deployment, and maintenance. Each phase ends with a Milestone 1 review which, if successful, indicates that the problem solution is possible and it is feasible to continue the effort.

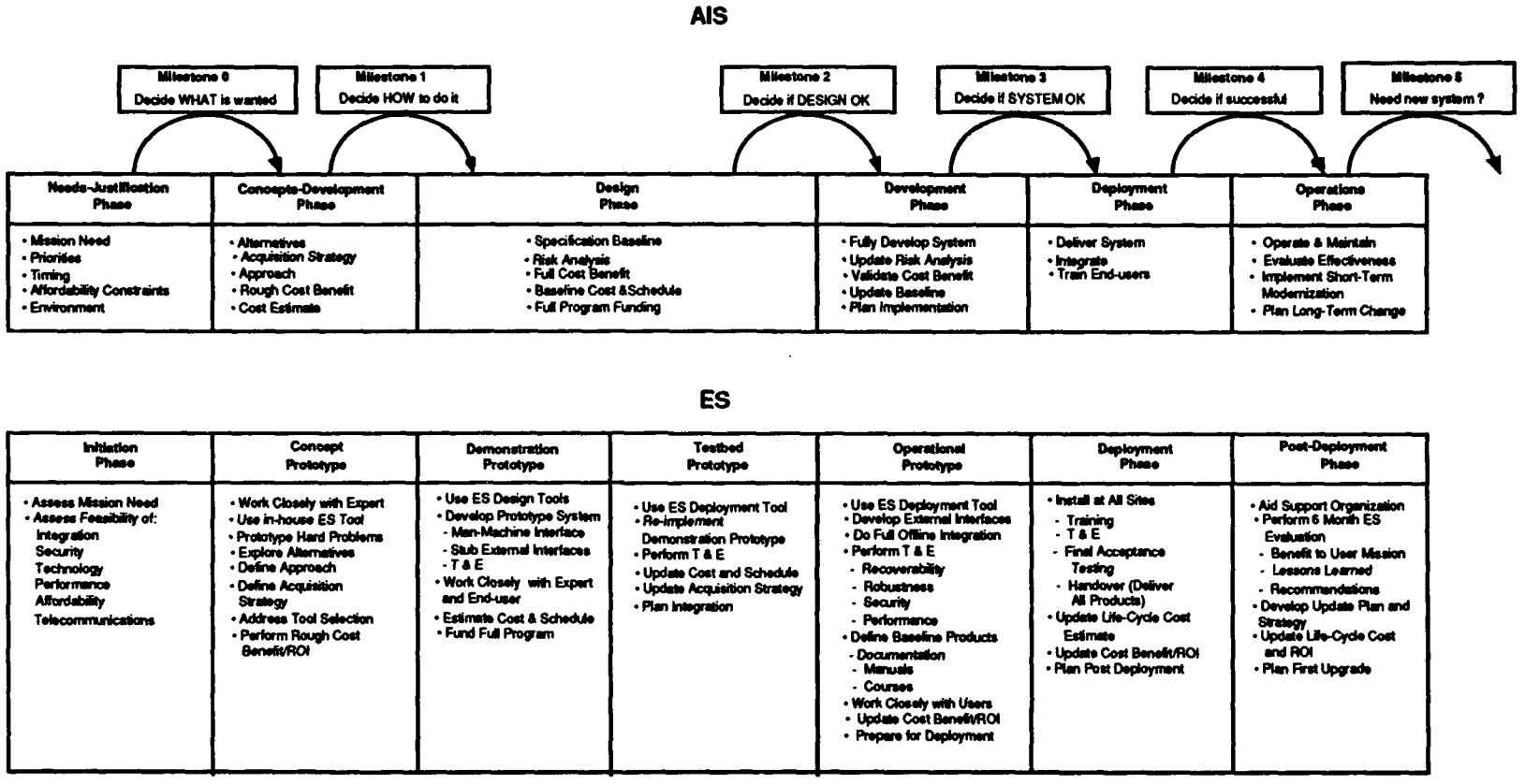


Figure 3.1—Mapping from DoD Directive 7920.1 to the ES development life-cycle model

Design Phase. The purpose of the AIS Design Phase is to complete the AIS technical specifications and validate the selected system design. This maps into the ES Definition/Design Phase which includes two prototyping efforts: the Demonstration Prototype and the Testbed Prototype. The Demonstration Prototype focuses on rapid prototyping to explore, define, and demonstrate the problem solution in a stand-alone mode with stubbed-in interfaces to required AISs. The Testbed Prototype is a redesign and re-implementation of the Demonstration Prototype based on the ES tool that will be used in deployment. Both the AIS and ES efforts address demonstration and prototyping activities, maintenance and logistics requirements, development of security specifications, and the resources needed to satisfy the requirements of the program plan and proposed schedule. The main difference in the efforts is that the AIS Design Phase results in a baseline specification while the ES baseline is established in the next phase (the Development Phase). The AIS Design Phase, and the ES Demonstration Prototype and ES Testbed Prototype each end with a Milestone 2 review which, if successful, indicates the design is acceptable and it is feasible to continue the effort.

Development Phase. The purpose of the AIS Development Phase is to develop the automated information system, test the completed system to ensure that it satisfies mission needs, and prepare for deployment. This maps to the ES Development Phase Operational Prototype, which is an integration of the Testbed Prototype into an example operational environment. Both the AIS and ES efforts address full-scale development; operational testing of the system validated against user requirements; plans for deployment, training, operations, maintenance, logistics support, and continuity of operations; and the resources necessary to satisfy the requirements of the program plan and proposed schedule. Each phase ends with a Milestone 3 review which, if successful, indicates that system implementation is acceptable and it is feasible to continue the effort.

Deployment Phase. The purpose of the AIS Deployment Phase and the ES Deployment Phase is to field the respective systems in accordance with the approved deployment plan. Both efforts address resources needed to satisfy the requirements of the program plan, the proposed deployment schedule, and full operations and maintenance; management transition to the operational organization; and procedures for collecting and evaluating benefits, correcting system malfunctions, responding to functional user needs, and assuring continuous use of approved security safeguards. Each phase ends with a Milestone 4 review which, if successful, indicates successful deployment and continuance to the next phase.

Operations Phase. The purpose of the AIS Operations Phase and the ES Post-Deployment Phase is to operate and maintain the automated information system; evaluate AIS effectiveness and benefits; implement the approved short-term post-deployment modernization plan; and plan for long-term existing AIS modernization. The AIS and ES efforts end with Milestone 5 reviews to determine if the existing system continues to satisfy validated mission needs, requires modernization, or should be terminated.

The above discussion has shown a straightforward mapping from the AIS life-cycle management process to the ES life-cycle management process. The next section discusses ES life-cycle documentation requirements with respect to DoD Directive 7920.1.

3.2 ES DOCUMENTATION REQUIREMENTS WITH RESPECT TO DoD DIRECTIVE 7920.1

Software documentation standards are generally narrow interpretations of more flexible models. While the underlying model might lend itself to certain types of functional documentation, the standards, not the model, actually establish the number, format, and content for those documents; they regulate when those documents are prepared; and, they may also stipulate when documents are to be baselined. Such documents generally exist (1)

to provide information required to support management decisions at key transition points in the development life cycle and (2) to assist the maintainers of software.

For example, DoD Standard 7935.1 describes 13 major documents to be prepared and used during software development and refers to those documents by two-letter abbreviations as shown in Table 3.1. (Short descriptions of the documents can be found in Appendix B.)

Table 3.1

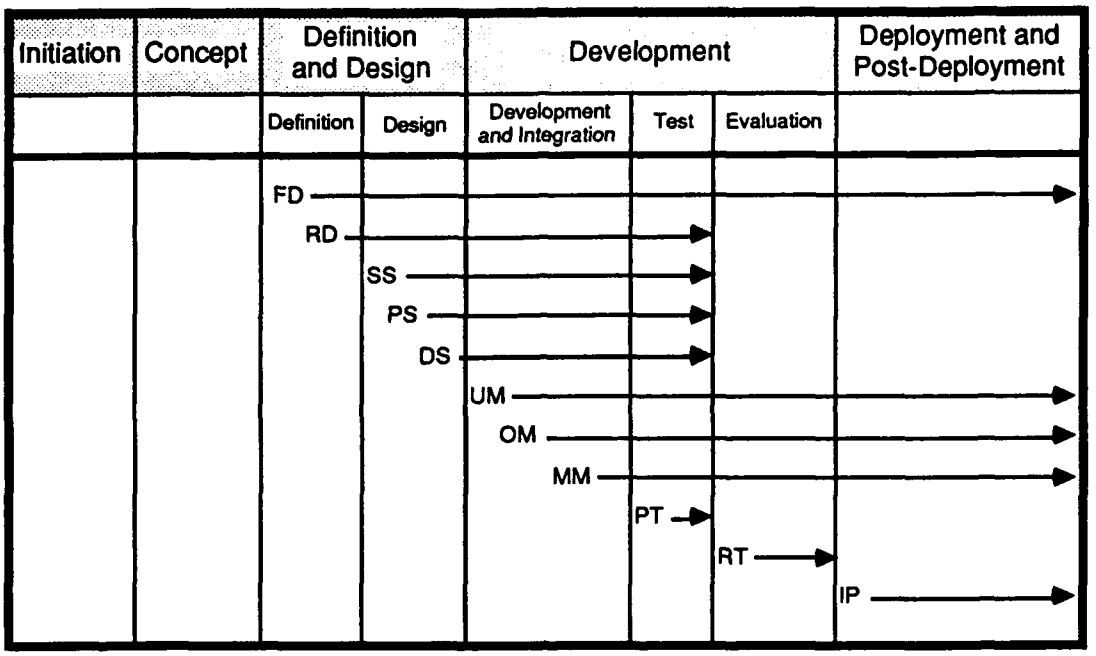
DoD DOCUMENT ABBREVIATIONS

Abbreviation	Definition
FD	Functional Description
SS	System/Subsystem Specification
US	Software Unit Specification
OM	Computer Operation Manual
DS	Database Specification
UM	Users Manual
EM	End User Manual
MM	Maintenance Manual
PT	Test Plan
RT	Test Analysis Report
IP	Implementation Procedures
RD	Data Requirements Document
PS	Program Specification

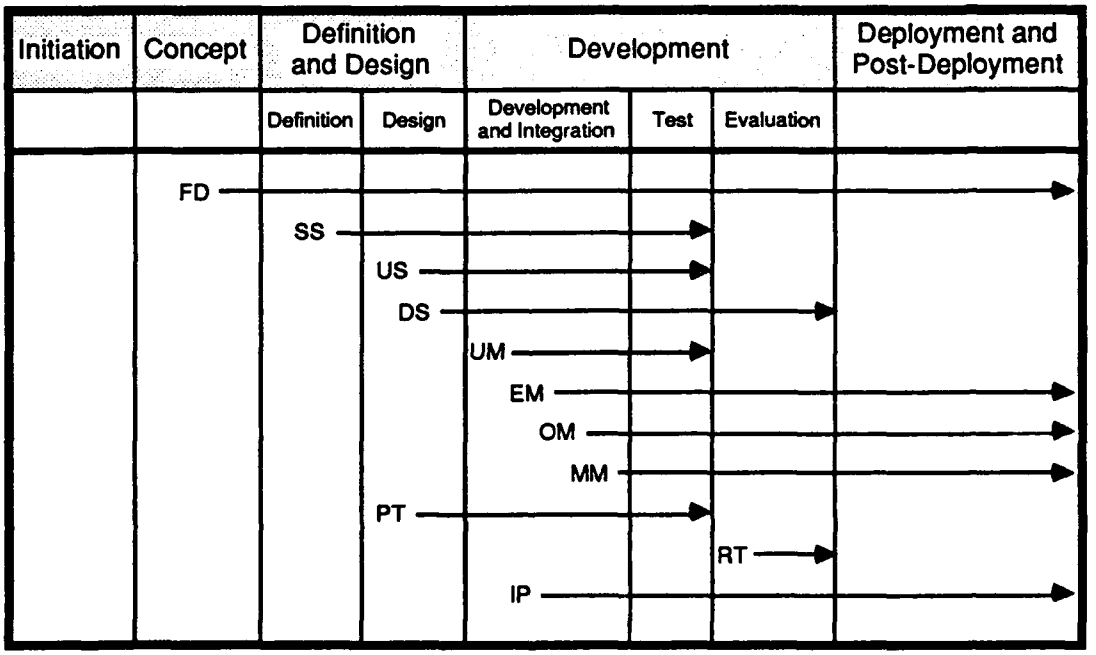
Figure 3.2 shows the standard DoD documents prepared and used across several phases of development. The two-letter abbreviation is shown at the point where the associated document is initiated. An arrowhead indicates the point at which that document is finalized and no longer subject to revision. These figures are extracted from the old and new versions of DoD Standard 7935.1, respectively, and reflect a changing attitude as to when the importance of certain documents (e.g., the FD, SS, PT, and IP) begins. The supporting narrative in draft DoD Standard 7935.1 avoids a commitment as to where document preparation ends and use begins. Hence, it is acceptable within the framework of this standard for a document such as the FD or SS to be updated throughout the life of a project without ever being baselined.

There are three ways in which ES documentation needs differ from those of conventional DoD software development efforts. First, the conventional DoD demand for early specification and baselining is inconsistent with ES life-cycle development. Second, current DoD documentation standards make it difficult to (1) document the functional contributions of ES support tools (such as an ES shell) and (2) to deal with the categorical split of documentation into data and procedures especially when describing ES constructs that are a combination of both. Last of all, a new document, the Progress Notebook (PN), is required to support special ES documentation needs.

Problems Created by Early Specifications and Baselining. In spite of the open-endedness of 7935.1 with respect to document completion, strict baselining and early demand for elaborately detailed specifications are typical of U.S. Government contracts for conventional software systems. This demand for fully enumerated documents, such as the functional and systems specifications, early in the life-cycle process is detrimental to ES development, where the various prototypes themselves become part of the problem definition. Expert system design decisions and functionality built into each increasingly complete prototype contribute to the final design and system specification. Any management approach that prescribes premature baselining and overly detailed specification early in a project will almost always result in expensive documentation that adds to the time line but which



Document Preparation and Use per DoD Standard 7935.1, April 24, 1984



Document Preparation and Use per DRAFT DoD Standard 7935.1, January 15, 1988

Figure 3.2—Automated information systems life-cycle documentation

quickly becomes obsolete. This problem can be overcome if ES developers are allowed to prepare and deliver the standard documents incrementally throughout the development life cycle. Updated at the transition points between prototypes and phases, each document should describe the system through its current stage (to some reasonable degree of thoroughness) and only refer generally to stages further along in the life cycle. This approach is consistent with DoD Standard 7935.1, as described above.

Contracted software acquisition is, by nature, less flexible than in-house development. A review of case studies revealed several poignant examples of an almost crippling impact on ES development efforts when contracts demanded extensive, detailed documents at points which were out of synchronization with the ES life-cycle model. Rome Air Development Center (RADC) recently received the results of a study that recommended modifications to the DoD procurement processes in order to specifically accommodate the acquisition of AI software [Bardawil 1987]. Barry Boehm points to recent progress in "establishing more flexible contract mechanisms, such as the use of competitive front-end contracts for concept definition or prototype fly-offs, the use of level-of-effort and award-fee contracts for evolutionary development, and the use of design-to-cost contracts" [Boehm 1988, page 70]. He also notes that acquisition managers need to feel comfortable using the new contracting mechanisms. Our recommendation to an ES manager is to work out the software acquisition process with his/her contracting agent to allow for incremental development and for deviation from standard DoD document and baseline agendas.

Problems with Applying Current DoD Documentation Standards to ES Development. Most software documentation standards emphasize early 1970s hardware requirements rather than approaching the documentation from a software perspective. This complicates the programmer's job both during development and when trying to reconstruct a view of software using such documentation. For ES developments two particular problems are caused by documentation that is hardware-driven: (1) documenting functionality contributed by ES support tools, and (2) dealing with the categorical split of documentation for data and procedures.

Documenting functionality acquired from ES support software tools (i.e., shells) is difficult. Some ES development tools are provided with source code in order to improve programmer productivity by allowing for wholesale cannibalizing of code. Some tools provide libraries of routines, which can be used by ES developers to provide specific common functionalities. All tools lend themselves to certain types of applications and should be chosen because of specific capabilities they provide. Thus, the development tool actually contributes functionality to the final ES design and should be considered as part of the system specification. The latest revision to DoD Standard 7935.1 addresses this contribution by providing pointers to appropriate documents and paragraphs within documents based on the nature of the functionality borrowed from a support software tool. Table 3.2 provides guidance based on that standard. While this is a step in the right direction, it can lead to documentation which is repetitive, difficult to write, and difficult to follow.

Documentation that describes data in one section and procedures in another is inappropriate for documenting certain ES components. Object-oriented programming, for example, is a non-procedural method of programming which evolved out of simulation and modeling. In object-oriented programming, both data and procedures are bound together into elements called objects. In response to messages passed to them, objects may take action, send a message to other objects, or be acted upon. Internally, objects are both data and procedures. They are typically implemented using frames, a data structure in which the various attributes pertaining to the individual object are stored in slots. One or more of those slots could contain procedures, called methods, which define both the domain of messages the object can send or receive and the executable logic used to respond to or initiate messages. Standard DoD documentation typically requires data and procedures to be described in separate and specific formats. While this separation of data and program specifications is reasonable for statically structured languages if one is interested in separately assessing the impact on CPU and storage resources, it requires a bit of mental

gymnastics for programmers who are trying to describe how an object functions in their object-oriented programs. This Guide recommends that the project manager decide whether such ES constructs will be documented in the FD and SS as data or procedures, note this decision early in the documents, and state the decision again in both the data and function areas of the documents.

Table 3.2

APPROPRIATE DOCUMENTS TO DESCRIBE SUPPORT SOFTWARE TOOLS

	Document								
Document Tool Function	FD	SS	US	DS	OM	MM	PT	RT	IP
If it aids in stating and recording requirements	X								
If it is used in operation of the ES	X	X					X		
If it supports test and evaluation, development of test data, review and audit of tests, or test operations	X				X				
If it supports implementation, conversion, or interface to existing automated information system	X	X							X
If it aids in the overall design and development		X				X			
If it aids in design and development of specific software units	X	X	X			X			
If it aids in the review of test results or audit of the expert system								X	
If it enhances capabilities or performance of database				X					

A New Document: The Progress Notebook. The ES life-cycle process will require the documents shown in Figure 3.2 for draft DoD Standard 7935.1, January 15, 1988. In addition, in order to support special ES documentation needs, a new document, the Progress Notebook, should be initiated by the project manager at the start of the Initiation Phase.

The purpose of the Progress Notebook is to document the history of all relevant project activities, some of which might otherwise never be permanently documented, remaining only in the memories of the development team members.

In principle, the PN is a living document, an historic trace of all alternatives and decisions, open issues, blind alleys pursued, and the like. It fills in the gaps left by the standard documents and serves as a navigational aid for reading through the other project documents. The PN contains a wide variety of information with varying degrees of sensitivity and serves as a central source of information for the project team as well as for future maintainers of the resulting ES.

For ease of reference and to avoid duplication, the PN should include the following standard documents:

FD Functional Description

SS System/Subsystem Specification

US Software Unit Specification

DS Database Specification

PT Test Plan

RT Test Analysis Report

IP Implementation Procedures

In addition to the standard documents, the PN should also include the following useful information:

- List of those involved: people in the user, developer, automatic data processing, security, communications, training, and maintenance organizations as well as all people consulted about the project.
- Notes from meetings: dates, attendees, subjects covered, action items, and critical information related to issues discussed, decisions made and their rationale, project organization decisions, discussions and evaluation of risk factors.
- Transcripts from knowledge engineering sessions and notes on the representation strategies considered and accepted, explaining the mappings from the expert's domain knowledge to the form implemented in the software.
- Descriptions of briefings: dates, purposes, attendees, copies of material presented, results of briefings.
- Documentation of information collected during the technical tasks (descriptions of the problem, problem environment, written data/knowledge sources, user community, management attitudes about the problem and ES technology, and all identified risks in these areas).
- Documentation of the risk analysis, prioritization of risks and recommendations for risk resolution; feasibility analysis and recommendations; notes or references to similar ES products or development efforts; and the rough cost estimate.
- Project reviews: dates, attendees, copies of materials presented, review results including all action items, and dates for follow-ups (in particular the milestone reviews).
- "Problem" reports and recommendations as a result of the prototype development and T&E.

The PN should be structured to facilitate broad access to its contents by the team members without compromising its more sensitive components (e.g., contractor reviews or internal product evaluations). In the best of all possible situations, the PN should be on-line (perhaps utilizing hypermedia techniques) to support multiple views to meet the varying needs of the project team and their management. Lacking the facilities to maintain the PN on-line, the project manager should keep it in a well-organized binder with plenty of room for insertion and expansion.

The next section discusses three major themes important to ES life-cycle development.

3.3 MAJOR THEMES IN ES LIFE-CYCLE DEVELOPMENT

Three major themes in ES life-cycle development are: (1) the importance and possible difficulties associated with integration, (2) the need for continuous user and domain expert participation throughout the ES life cycle, and (3) the need for dynamic documentation of the design and development processes.

Integration. Currently, the majority of expert system efforts have been stand-alone systems, many of which have not proceeded past the Definition/Design Phase. Many of these ESs require data that reside on AISs but have finessed integration issues by (1) having users key in the data when needed or (2) integrating a static data file, representative of the AIS data, into the ES. Most future ES applications will not be stand-alone applications but will need to integrate with the rest of the information processing world through electronic interfaces.

For ES technology to be effectively used, integration issues must be continuously addressed from the Initiation Phase through the Post-Deployment Phase. Continuity is especially important because many AISs may be undergoing modernization updates during the development of the ES; an integration approach defined in the Concept Phase may not work for the Operational Prototype. Integration concerns cannot be put off until the Operational Prototype because their solution will have a strong effect on the system architecture, tool selection, cost, schedule, and risk. The project manager must be continuously watchful for changes in AISs, other programs or software packages that the ES will interface with (e.g., spreadsheets, database systems), I/O devices, communications networks, security requirements, etc., that could affect ES integration. He/she must address integration at every in-process review and milestone review.

Participation of Relevant Communities. Expert system development is a people-intensive activity requiring participation from the user, ADP, security, relevant AISs, communications, training, and maintenance communities. Expert System development requires continuous user and domain expert participation during the entire life-cycle process. The user community must commit to providing this support as a prerequisite to undertaking an ES development effort. Likewise, the participation of relevant experts in ADP, security, communications, training, and maintenance should be assured. In particular, representatives from AISs relevant to the expert system should participate and be made aware of all relevant ES interface decisions. Likewise, the ES project manager should be made aware of all AIS plans and changes that would affect the interface between the ES and the AIS. The project manager has a responsibility to continuously seek feedback from the people in these various organizations as to how the effort is proceeding and where problems may exist. He/she should invite those whose areas of expertise are relevant to the in-process and milestone reviews.

Dynamic Documentation. The Guide continuously reminds both project managers and technical people of the need to update the Progress Notebook and other relevant documentation throughout the life-cycle development process.

3.4 ORGANIZATION OF THE GUIDE

The first three chapters have introduced the Guide and have shown how ES life-cycle development can be mapped to DoD conventional software life-cycle development and to standard DoD documentation requirements. Each of the remaining chapters discusses a phase (or prototype effort) in the ES development life cycle and describes how the effort should be addressed. Conventions have been used to make the chapters consistent and the Guide easy to use.

Each chapter has a management track and a technical track and is divided into the following four sections:

- x.1 Introduction
- x.2 Risk Containment
- x.3 Management Activities
- x.4 Technical Process

The "Introduction" contains two important figures. Figure x.1 shows the expert system development process (as shown in Figure 3.3) with the appropriate box darkened to show the phase being addressed by the chapter. Figure x.2 folds out and is a critical roadmap to the chapter and a process chart to the development phase being addressed.

Figure 3.4 is an example figure taken from Chapter 4. The dotted horizontal line in the middle separates the management activities on the top half of the chart from the technical process on the bottom half of the chart. The major boxes are labeled with an "M" followed by a number for management activities or by a "T" followed by a number for technical activities. If an activity is a joint management and technical activity, its box intersects the dotted line and is labeled with a "J" followed by a number. The labels are referred to in Section x.2 and Section x.3 (i.e., they serve as index points). The Introduction contains a synopsis of the two tracks of activities that can be more easily understood if read in conjunction with Figure x.2.

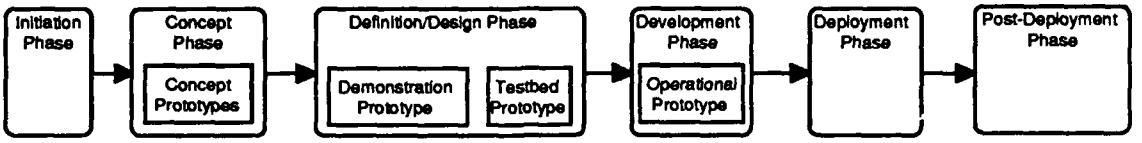


Figure 3.3—Expert system development process, sample figure

The "Risk Containment" section is a single paragraph that refers to Table x.1, which is an example prioritized list of the top ten risks for the phase addressed by the chapter. Obviously, risk evaluation depends to a large extent on the application. Table x.1 is meant to suggest types of risk that may be relevant to the phase, and it should be read as an example not a recommendation.

Section x.3, "Management Activities," begins with an unnumbered introduction that addresses relevant management issues for the phase covered by the chapter and lays out the pertinent subsections.

Section x.4, "Technical Process," begins with an unnumbered introduction that addresses relevant technical issues for the phase covered by the chapter and lays out the pertinent subsections.

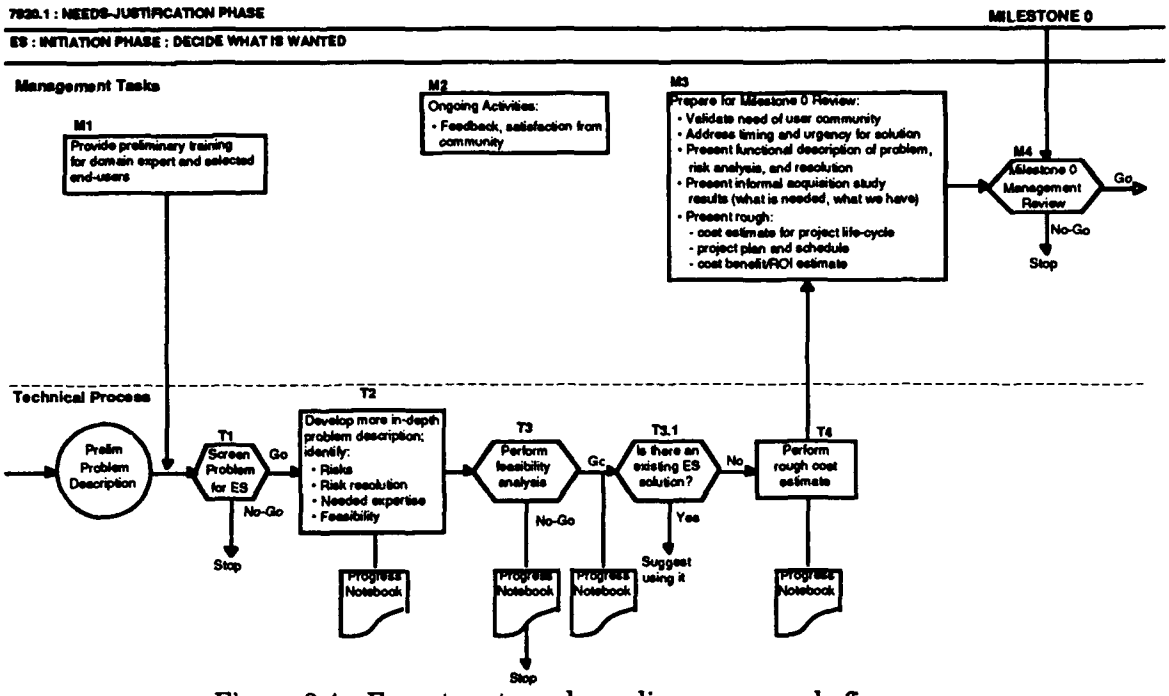


Figure 3.4—Expert system phase diagram, sample figure

4. INITIATION PHASE[2]

4.1 INTRODUCTION

Figure 4.1 shows the place of the Initiation Phase in the expert system development process. This phase is concerned with "deciding what is wanted," and its major management and technical activities are shown in Figure 4.2.

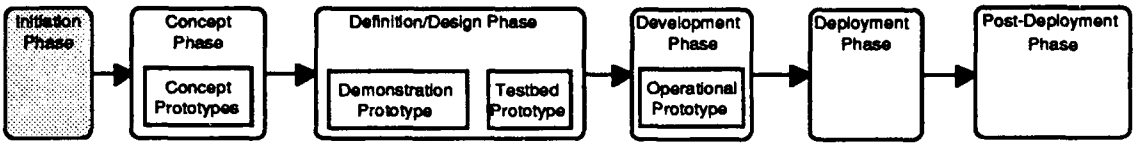


Figure 4.1—Expert system development process, Initiation Phase

Management tasks include setting up an interim project plan for the phase; interfacing and arranging for meetings with personnel in the user, ADP, and maintenance organizations as necessary; managing the technical studies; and using the results of the technical studies to prepare for the Milestone 0 review. The final management activity in the Initiation Phase is to present the case for an ES development effort at the Milestone 0 review.

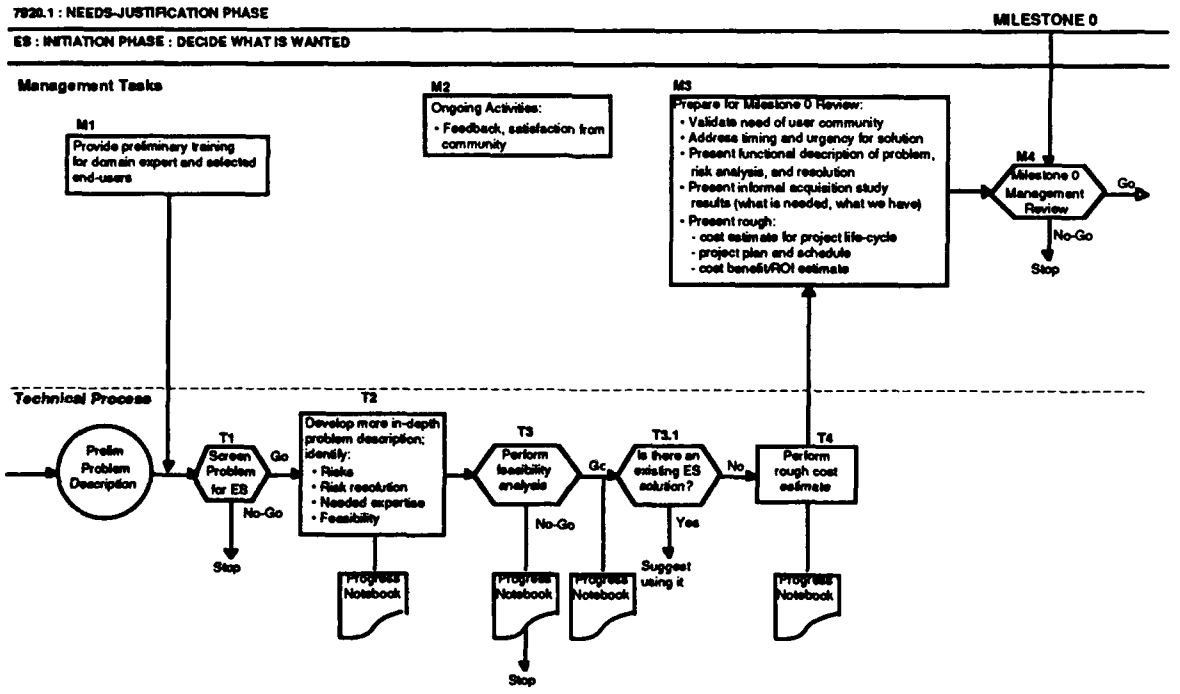


Figure 4.2—Initiation Phase

[2]See Appendix A for definitions of acronyms and terms used in this chapter.

The technical process begins with an initial screening of the problem to quickly reject a problem not suited to an ES solution. For a problem passing initial screening, a more in-depth examination of the problem is made, which includes collecting information for a feasibility analysis to determine that the problem is: appropriate, of acceptable risk, of valid concern, worth solving, and assured management support. After the in-depth examination, a go, no-go decision is made. The technical studies are documented in a Progress Notebook (PN) including a risk analysis and risk prioritization and a rough cost estimate, which are used in preparing for the Milestone 0 review.

Throughout this document the periodic reviews recommended are described in terms related to the "Major Automated Information System Review Council (MAISRC)" process used to review very large (greater than $100 million) programs [MAISRC 1987]. For smaller efforts the managerial level at which reviews will be done should be determined during the Initiation Phase, based on project complexity.

This chapter is divided into four sections: Section 4.1, this section, is an introduction; Section 4.2 presents a risk containment table; Section 4.3 describes the management activities; and Section 4.4 describes the technical process.

4.2 RISK CONTAINMENT

Table 4.1 is an example prioritized list of items of major risk to the Initiation Phase that need to be identified and addressed during that phase.

4.3 MANAGEMENT ACTIVITIES

The management activities are organized into:

4.3.1 Initial Management Activities (M1)
4.3.2 Ongoing Management Concerns (M2)
4.3.3 Preparing for the Milestone 0 Management Review (M3)
4.3.4 Holding the Milestone 0 Management Review (M4)

4.3.1 Initial Management Activities (M1)

The initial management activities are: to set up an interim project plan, to acquire necessary technical staff to perform initiation tasks, and to define the documentation needs for the project, in particular the needs during this phase.

The interim project plan includes cost and schedule for the major technical tasks: screening the problem, investigating the problem in-depth, identifying and resolving risk, analyzing feasibility, and performing a rough cost estimate. The plan should include scheduling a Milestone 0 review.

Of extreme importance in this phase is a top-talent Knowledge Engineer (KE) or analyst to perform the technical tasks. He or she will be likely to recognize problematic areas—no matter whether they have been successfully or unsuccessfully addressed in the past—thus reducing a large amount of exploratory effort (possibly leading to blind alleys) in the Concept Phase. The ability to quickly identify insolvable aspects of the problem and advise early termination of the effort will also reduce costs.

To support ES documentation needs, the Progress Notebook should be initiated by the manager at the start of this phase.

4.3.2 Ongoing Management Concerns (M2)

Ongoing management concerns include: facilitating exchanges across communities, maintaining the schedule, and soliciting project support.

Table 4.1

AN EXAMPLE PRIORITIZED LIST OF INITIATION PHASE RISK ITEMS

	Risk Item	Risk Management Technique
1.	Personnel shortfalls	Staff this phase with top-talent analyst or KE; plan to acquire staff for next phase
2.	Lack of cooperation and participation across communities (developer, user, ADP, maintenance)	Discuss need for team effort; explain need for ongoing user community commitment and enthusiasm in particular; identify credible domain expert(s), end-users, and test cases
3.	Straining of ES capabilities	Scope application to plan to explore doable solutions in Concept Prototypes
4.	Difficulty integrating ES with AIS	Explore existing solutions; explore feasible low-risk designs; be sure to address security issues; get ADP support for evaluating options
5.	Validity of problem solution to mission	Determine that solving the problem is of high importance and high visibility
6.	Need for secure or multilevel secure system	Explain that lack of technology to support multilevel security means ES and AIS will need to run at system high; define computer, data, system, physical security needs, and explore ways of satisfying them within DoD Security Regulations
7.	Inadequate funding, unrealistic schedule and expectations	Identify champion and discuss ES technology and reasonable expectations to aid in acquiring funding and in supporting project goals
8.	Lack of maintaining the Progress Notebook	Explain to development staff the necessity of keeping history of project: (1) to document explorations, decisions, etc., that may need to be re-examined or re-justified in the future, (2) to facilitate writing of documents required by 7935.1 and (3) to educate future staff; be sure currency of Progress Notebook is part of management and technical objectives
9.	Difficulty in doing costing	Examine similar completed ES efforts in P&L, DoD, and the commercial world
10.	Difficulty in doing cost benefit/ROI	Examine similar ES examples; increased capabilities may be easier to describe than quantify; enhanced productivity is even harder; no easy answer

The manager should facilitate information exchange across user, developer, ADP, and maintenance communities by setting up meetings, explaining the need for a team approach to ES development, and offering seminars in ES (perhaps including a demonstration of a relevant ES). Of utmost importance is securing the enthusiasm and commitment of the user community, including management. Credible and available domain experts should be identified and their participation, as well as that of end-users, agreed to. In addition, commitment to supplying adequate test cases is needed.

Cost and schedule should be maintained, but if hard technical problems arise the project manager should be prepared to adjust cost and schedule to allow for their further exploration. This is especially important if the issues are serious enough to make the project inviable. In the long run, this is a cost-saving approach.

As this phase progresses and the potential for a successful project increases, the project should be briefed to higher management in order to attract champions, generate enthusiasm, and ensure support.

4.3.3 Preparing for the Milestone 0 Management Review (M3)

The purpose of the Milestone 0 Management Review is to decide whether to continue the project. Information to be presented includes:

- A validation of the need of the user community for the problem solution.
- A discussion (if appropriate) of any urgent timing requirements of the user community for the problem solution.
- A functional description of the problem, its risk analysis, risk resolution, and management techniques.
- The results of an informal acquisition study to show that the resources needed to proceed to the Concept Phase can be met.
- A rough cost estimate of the project life-cycle costs with respect to planned funding and affordability constraints.
- A rough project plan and schedule.
- A rough cost benefit/ROI estimate.

Validation of Need. Information presented to validate the user community's need for the project should include the importance of the problem solution to: the mission of the user community, the users, the managers, and other organizations. Any information about previous ES products or developments designed to solve a similar problem should be presented, along with explanations as to why they are not a solution to the current problem and how this project is planning to benefit from those experiences.

Time Schedule. If the user community has a schedule or urgent requirement for the problem solution, present the requirement and address the risk (if any) in trying to meet that requirement.

Functional Description. All information about the functional description of the problem to date, risk analysis, and risk resolution should be derivable from the Progress Notebook. Management techniques for handling the risk items should be presented, particularly with respect to addressing hard technical problems in the Concept Phase.

Informal Acquisition Study. The informal acquisition study should briefly address life-cycle needs including:

- Development organization staffing needs;
- Required user community participation (e.g., expert(s) and end-users);
- Required ADP and maintenance organization participation;
- Training needs;
- Hardware and software needs;
- Integration (interfacing) needs.

A detailed plan of what is needed in the Concept Phase and how the project manager (PM) plans to meet these needs should be presented. This should include:

- Identifying the development staff members and any necessary training;
- Identifying the expert(s) and end-users that will participate in the Concept Phase and any necessary ES training;
- Identifying who in the user community will be responsible for furnishing the necessary test cases;
- Identifying who in the ADP and maintenance organizations will participate;

- Identifying the hardware/software tools that will be used in the Concept Phase and how they will be acquired;
- Identifying alternatives for contracting either development life-cycle support or consulting services, if applicable.

Rough Cost Estimate. There are basically two methods for determining the costs of an ES project: *cost the design* or *design to cost*. In *cost the design*, a rough cost for the project life-cycle is estimated based on the projected system specifications and the design information collected in the PN. It is advisable to base the cost estimate on anticipated level of effort, numbers of people involved, and other manpower costing data because there appear to be no reliable industry-wide metrics for costing the ES software itself (number of rules, number of facts or assumptions in the fact base, number of object classes, etc.). One of the best sources of costing knowledge comes from actual case studies of other similar ES projects. This sort of information may not be easy to obtain for commercially developed systems—because of its proprietary nature this information is rarely included in printed reports and is generally closely guarded. Systems developed for the government, on the other hand, are open for closer inspection. Costing data can be gleaned from proposals, progress reports, and from conversations with project managers.

In *design to cost*, management should be prepared to set a funding ceiling within which the ES development team must work. The cost estimate should be consistent with any known or planned funding constraints. If cost constraints will affect the functionality of the problem solution, then demonstrate that the task can be reduced in scope and that the solution will still be valid to the user community. If the task must be reduced to accommodate cost, then the project plan should be likewise reduced. As a development project progresses, increased functionality can be added if money remains in the budget; otherwise, management and the user community must be willing to accept the lower level of functionality and capability in the delivered ES product. Many commercial systems appear to have been costed in just this manner. Costing the design is often considered too difficult or unreliable to use because of the lack of detailed specification early in the ES project life-cycle and because of a general lack of experience fielding expert systems; designing to cost is considered a very attractive alternative.

Rough Project Plan. A rough plan and schedule for the project life-cycle should be developed consistent with the rough cost estimate. The schedule is necessarily rough due to the nature of ES development, which uses prototyping and iterative refinement techniques extensively to determine functional requirements, design, and implementation techniques.

Rough Cost Benefit/ROI Estimate. A rough cost benefit/ROI estimate may be difficult to do but should be developed if possible. In general, the benefits which accrue from the use of ES technology are in one of two categories: productivity enhancement or increased capability. Of the two, productivity may be easier to evaluate in certain situations. For example, some ESs might capture the expertise of very valuable and highly paid personnel and place that capability in the hands of a smaller number of lesser-skilled personnel. This can be assigned a dollar value, but the cost of keeping up the knowledge base (especially for an application where the rate of knowledge change is high) may be difficult to determine (i.e., there are so few fielded ESs from which to gather experience).

Many ES applications fall into the category of providing increased capabilities. Some applications may allow us to do jobs that we could not do before. Since the job is not currently being done, its real value to the organization will be hard to quantify unless some sort of direct financial benefit is obtained from applying the new capability. In the U.S. Government, particularly DoD, such immediate financial benefit does not usually exist; rather, the immediate benefit is often an enhanced mission capability. While such a benefit is attractive on the surface, and the analyst might be convinced of its immense value to the organization, the inability to assign a dollar value to it makes calculating ROI difficult.

4.3.4 Holding the Milestone 0 Management Review (M4)

The last management activity in the Initiation Phase is to present the project at the Milestone 0 Management Review. We strongly recommend that managers at all levels of the various participating organizations be encouraged to attend, as well as expert and end-user representatives from the user organization.

4.4 TECHNICAL PROCESS

The technical process in the initiation phase consists of four tasks, each of which is addressed in a separate section below.

4.4.1 Initial Screening (T1)
4.4.2 Further Analysis of the Problem (T2)
4.4.3 Results of Feasibility Analysis (T3)
4.4.4 Rough Cost Estimate (T4)

4.4.1 Initial Screening (T1)

The objective of the initial screening is to cursorily identify whether the problem is an appropriate project choice, and, if so, whether ES technology is an appropriate solution choice. Of major concern are: whether the user community organization and development organization would offer sufficient commitment and support for the problem solution; whether the problem is worth solving in the sense of being a valid and important need of the user community in satisfying its mission; whether the problem is solvable using ES technology; and whether it might be better solved using a conventional software approach.

This process should generally take a few days to a week. During that time an analyst or Knowledge Engineer will discuss the problem with one or more domain experts or end-users, may observe the problem being solved, and may examine materials describing the problem or problem domain. The analyst will be seeking answers to questions about critical characteristics of the problem, the problem environment, the written knowledge sources, and the user community.

Critical problem characteristics:

- The problem and solution must be well defined.
- The problem must be solved mainly by heuristic or rule-of-thumb reasoning rather than by mathematical algorithms (though parts of the problem solution may involve the use of algorithms or arithmetic operations).
- The problem solution must not require the system to perform common sense reasoning or depend on general knowledge about the world. However, the system may be designed so that common sense is furnished by the interactive user.
- The problem should be neither too easy nor too hard to solve. A suggested guideline is that it should take one person on the order of one to twelve hours to solve [Winston 1987].
- The scope of the problem must be broad enough for the solution to yield useful results but narrow enough to make the project doable.

Critical environment characteristics for an embedded ES or an ES interfaced to an information system:

- The state of the embedding or interfacing system at the estimated time of deployment should be known and well defined. This includes security, communications, and defining all interfaces (e.g., software to software, software to hardware, machine to machine). If these systems exist, then risk is substantially reduced.

- Data used by the ES and to be furnished by another system should be: well defined, relatively error-free, and of reasonable size and flow. Test data should be available for ES development use.
- ADP technical staff should be available to participate in the evaluation of alternative designs for embedding/interfacing the ES.

Critical characteristics for written sources of data / knowledge:

- Data and knowledge to be derived from written sources should appear to be manageable in size and complexity.
- Data and knowledge taken from different written sources must be considered to be generally consistent, or there must be known methods for handling inconsistencies.
- Security classification levels of the data/knowledge should be known.

Critical community characteristics:

- Articulate and enthusiastic expert(s) and users should be available and committed to the effort as necessary during the project's life cycle.
- Multiple credible experts should be in general agreement about the solution.
- Management should consider the problem solution to be important to the organization's mission.
- Manager(s) should be enthusiastic, supportive, and understand the resource commitment and team effort (of user community, developers, ADP staff, and maintenance community) required to make the effort successful.
- Financial support should be available and considered adequate.

After evaluating the above characteristics, the analyst may decide that (1) the problem is not a good project choice, (2) the problem is a good project choice, (3) the problem is not amenable to an ES solution, (4) the problem may be amenable to an ES solution but risks exist requiring further investigation, or (5) the problem is a good candidate for ES technology.

If a problem is amenable to ES technology, the use of the technology must be justified. "Just because it's possible to develop an expert system for a particular task doesn't mean it is desirable to do so" [Waterman 1986]. Some acceptable reasons for justifying the use of ES technology are:

- Human expertise is being lost (e.g., through normal career moves or retirement).
- There are too few trained people with the necessary level of expertise.
- The complexity of solving the problem is becoming too great for individuals to master and perform within reasonable constraints.
- There is a long learning curve for a novice worker to become an expert.
- Expertise is needed in many locations.
- Expertise is needed in an environment hostile to human presence.
- It is desirable to standardize the decision process.

In summary, a problem that justifies the use of the technology and has an acceptable profile of characteristics is a good candidate for ES technology. For an application passing the initial problem screening, a Progress Notebook should be started and findings documented in it.

4.4.2 Further Analysis of the Problem (T2)

Once the problem has passed the initial screening, the KE or analyst begins a more in-depth investigation in order to define the problem. The goal of this activity is to acquire enough of an understanding of the problem, the scope of the solution, resources required, areas of expertise needed, and risks, to be able to validate the use of ES technology and form the basis for convincing management to support the project effort.

The exploration addresses five areas: problem definition, description of the problem environment, description of the written data/knowledge sources, description of the user community, and management attitudes about the problem and ES technology.

Problem Definition. A risk to be considered in using current ES technology is that users and/or developers often expect more from ES technology than can be delivered in an ES product. ES software is still immature. The KE or analyst should keep in mind the need for the solution to be as straightforward and simple as possible using successfully demonstrated ES capabilities. The KE should avoid considering capabilities that are still research issues (e.g., knowledge-base consistency, truth maintenance).

To develop the problem definition, the KE or analyst needs to:

- Become familiar with the problem domain and vocabulary by:
  - Reading documents; attending briefings, courses, and seminars; and having discussions with users.
  - Interviewing developers of similar projects.
- Work closely with a domain expert to develop a detailed operational description of the problem. This should include discussions and actually observing the expert at work. The KE should do the following:
  - Step 1: Observe the expert solving a typical case in one problem category.
  - Step 2: Follow with a discussion in which the expert analyzes why and how he/she made decisions.
  - Step 3: Formalize the analysis decision threads into reasoning steps and rule sets.
  - Step 4: Have the expert confirm the validity of the reasoning steps and rule sets.
  - Step 5: If applicable, incorporate suggestions of the expert to correct or deepen reasoning steps and rule sets, and repeat steps 1 through 5 for other cases in the same problem category.
  - Step 6: Repeat above steps with cases from a different problem category.
- Work with the domain expert to define the problem scope in such a way that it is broad enough for the solution to yield useful results but narrow enough to make the project doable.

The KE or analyst should identify problem definition risk factors such as:

- Areas of incomplete understanding or difficulty.
- Need to handle inconsistent data.
- Need for reasoning about spatial data (e.g., map or engineering drawings).
- Need for reasoning about temporal data (e.g., reasoning about events in the past, present, and future).
- Need for multiple users to concurrently access and update the knowledge base.

Description of the Problem Environment. Integrating an ES with a production information system is currently a risk because most commonly available ES tools have been built as stand-alone systems. Several tool builders have plans to implement or have recently implemented bridges to/from relational databases, but these have very limited functionality. Mainly, they do not manage data caching within the ES (data overflow is the ES developer's or user's responsibility), nor do they support the concept of a transaction (where data accessed from the "begin transaction" to the "end transaction" is consistent with respect to time). Expert system/database applications that are currently doable are those that are partitionable, i.e., where only a small set of data in the database is necessary for any one problem-solving instance.

The KE or analyst should:

- Describe the current production environment in which the problem exists (include relevant hardware/software, information system interfaces, interfaces to special devices, security, communications networks).
- Describe the future production environment in which the solution will exist (include relevant hardware/software, information system interfaces, interfaces to special devices, security, communications networks).
- If the future production environment is expected to change during the development of the ES, then include schedule dates and estimate the impact on the way of doing business.
- Describe current end-user terminal and workstation usage and plans.
- For each type of data needed for the problem solution, describe: data condition with respect to errors, security level, amount of data, flow rate of data, and data availability for use in ES development and testing.

The KE or analyst should identify environment risk factors such as:

- Risk factors related to system integration.
  - The ES is required to be embedded within or interfaced to an existing kludged system.
  - The ES is required to be embedded within or interfaced to a system undergoing change.
  - The ES will need to interface with a system that requires a different security level than that required by the ES.
  - The ES graphics will need to integrate with the resident support environment.
  - The ES will require support and configuration management of heterogeneous hardware, software, and databases.
- Risk factors related to data integration.
  - The ES needs to deal with erroneous data.
  - The ES requires more data than may fit in the workstation memory.
  - The ES requires a transaction capability for data acquisition.
  - The ES requires data that may arrive asynchronously and/or have processing time constraints.
  - The ES needs to use data at different security levels acquired from different sources.
  - The ES requires data for testing that may be difficult to obtain.

Description of Written Data/Knowledge Sources. The KE or analyst should:

- Identify written knowledge sources such as documents, manuals, directives, and regulations.
- For each written knowledge source, estimate:
  - Size of data and knowledge.
  - Complexity of the data and knowledge.
  - Frequency of data and knowledge format changes, and whether the ES must maintain old and new information structures.
  - Security level of data/knowledge.
- Obtain a credible expert's evaluation of the degree of consistency among the written knowledge sources.

The KE or analyst should identify risk factors related to written data/knowledge sources:

- Very large amounts of data may require an external DBMS, data caching techniques, transaction capability (as described under problem environment risks).

- Very complex knowledge may be difficult to capture, represent, maintain in a consistent way, or readily access in the ES.
- Frequent changes in format of data and knowledge may make it difficult and expensive to maintain the production system knowledge base.
- Inconsistencies among the knowledge sources may not be supportable with current ES technology.
- Data/knowledge from different sources at different security levels will require developing and operating the ES at a system high security level.

Description of the User Community. Since ES development requires a team approach, it is important to understand the user community, its resources, and possible working arrangements. The KE or analyst should describe:

- The user organization.
- Who the users and experts are and their job tasks in relation to the problem.
- Number of users and experts and their locations.
- The level of interest and enthusiasm of the users and expert(s) and their availability to contribute to the effort.
- The characteristics of a domain expert to work in close geographic proximity with the KE. The expert's knowledge and reputation must be such that if the ES is able to capture a portion of the expert's expertise, the system's output will have credibility and authority [Prerau 1985].
- User and expert skill levels with respect to: the domain, computer usage, and ES (e.g., users with little computer background will require different user interface capabilities and training from that of users with computer experience).
- Availability of sufficient test cases to support test and evaluation during the ES development.

The KE or analyst should identify user community risk factors such as:

- Low cost benefit because of very small user community or because the problem solution is needed infrequently.
- Unavailable and/or disinterested users and experts.
- Lack of a credible domain expert located close to the KE.
- Lack of sufficient test cases.

Management Attitudes Toward the Problem and ES Technology. The KE or analyst should describe community (user, developer, ADP, maintenance) management attitudes toward:

- The importance of solving the problem and the use of ES technology.
- The willingness to participate in a group approach to ES development.
- The need for adequate funding (e.g., a dedicated workstation with appropriate software for each KE, analyst, and expert).

The KE or analyst needs to identify management attitude risk factors such as:

- Management perceives the project as being of low importance.
- Management does not understand or cannot commit to the necessary level of support and involvement.
- Management has not approved sufficient funding.
- Management or circumstances would impose unrealistic time schedule.
- Management has unrealistic expectations about the results of applying ES technology.

The total collection of descriptive information should be documented in the Progress Notebook. Because we are describing a risk-driven approach to ES software development,

this is the first step in the process of identifying, analyzing, and prioritizing the risks, as well as documenting them with recommendations for their resolution and management.

The contents of the Progress Notebook, especially the risk analysis and resolution data, are the basis for the feasibility analysis, which is made by examining the benefit of the project in light of the identified risks and the cost of resolving the risks.

4.4.3 Results of Feasibility Analysis (T3)

The results of the feasibility analysis should be examined jointly by the user and developer communities in order to decide whether to continue the project effort. The feasibility analysis and issues and their resolutions (if there are any) should be documented in the Progress Notebook.

If the decision is made to continue the project, a serious effort should ensue to discover if any existing ES or conventional software system solves the problem. If such a system seems to exist, then the project should be directed toward using it. One or more additional feasibility studies may need to be made to determine whether any such existing systems offer a close enough fit to the problem solution so that cost would be lowered by modifying a copy of one of them rather than developing a new ES. In either case, the analysis, design issues, and lessons learned from the existing system(s) should be documented in the Progress Notebook for future reference.

4.4.4 Rough Cost Estimate (T4)

The last technical task in the Initiation Phase is to make a rough cost estimate of the proposed ES based on the information gathered (including knowledge of available funding), the feasibility analyses, and community experience. Community experience can be obtained by examining previous and current ES efforts in the P&L community, DoD, and the commercial sector. Information about efforts attacking problems similar to the one at hand will be the most useful. If such efforts are not available, then efforts similar in other respects should be examined including those having the same ES generic classification (e.g., diagnostics, planning), similar integration needs, and similar data/knowledge needs.

5. CONCEPT PHASE[3]

5.1 INTRODUCTION

Figure 5.1 shows the place of the Concept Phase in the expert system development process. This phase is concerned with "deciding how to solve the problem," and its major management and technical activities are shown in Figure 5.2. The gray boxes in Figure 5.2 represent the major technical steps in developing a concept prototype.

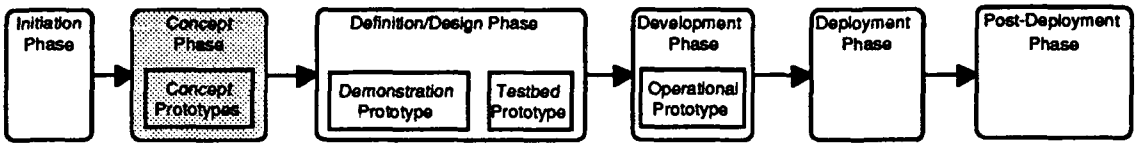


Figure 5.1—Expert system development process, Concept Phase

The objective of the Concept Phase is to develop a deep enough understanding of the problem, through concept prototyping and information gathering, to identify and resolve risks and define the scope of the problem to be developed in the Demonstration Prototype stage. Because conceptual prototyping is an exploratory process that begins with a problem that is only partially defined, the project plan describes rough goals to be accomplished within a specified amount of effort, time, and cost. As the work proceeds, unforeseen issues and needs will arise that will affect the plan. The Knowledge Engineer (KE) should keep the project manager (PM) informed, so that possible changes to the project plan can be considered and handled in a timely manner.

Management activities include: carrying out the project plan developed in the Initiation Phase; providing the necessary ES training to the user community experts and end-users who will be working with the KE; providing the necessary in-house tools for concept prototyping; facilitating interactions and exchanges across the developer, user, ADP, and maintenance communities; and developing life-cycle plans for quality assurance (QA), configuration management, security and integrity, test and evaluation (T&E), maintenance, training, and documentation. After the project manager has determined the adequacy of the functional description to date, an acquisition strategy is defined. Finally, the functional description (FD) including risk analysis, resolution, acquisition strategy, costing, and project plan is presented at the Milestone 1 review to demonstrate that the users' needs are still valid and the solution of the problem is well enough understood to proceed to the Definition/Design Phase.

The technical process begins with concept prototyping. It uses in-house tools to more deeply explore problem issues and potential risks in order to better define the problem space (defined in Section 5.4.1.1) for the Definition/Design Phase. Following and/or concurrent with concept prototyping, other application information is gathered along with the prototyping results into a checklist of characteristics. These are weighted and prioritized to form checklist profiles of the application characteristics and the ES tool capabilities needed for the Definition/Design, Development, and Deployment Phases. The adequacy of the FD is reviewed in-depth by members of all communities (development, users, maintenance, ADP, and security). If the FD is not adequate, the effort cycles back through the technical process until the FD is satisfactory. An acquisition strategy is then selected using the checklist profiles as a basis for exploring alternatives. When an acquisition strategy has been decided

[3]See Appendix A for definitions of acronyms and terms used in this chapter.

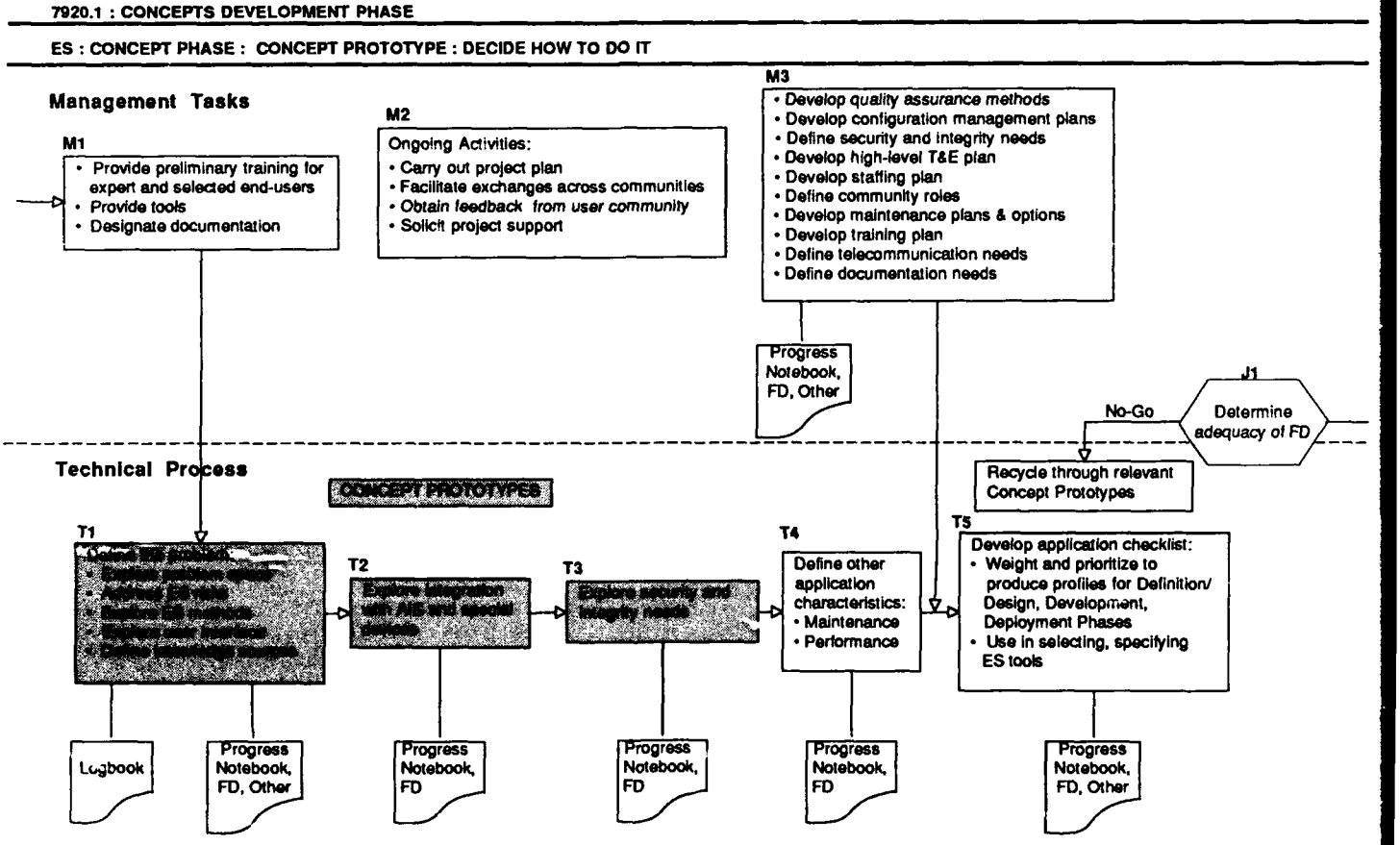
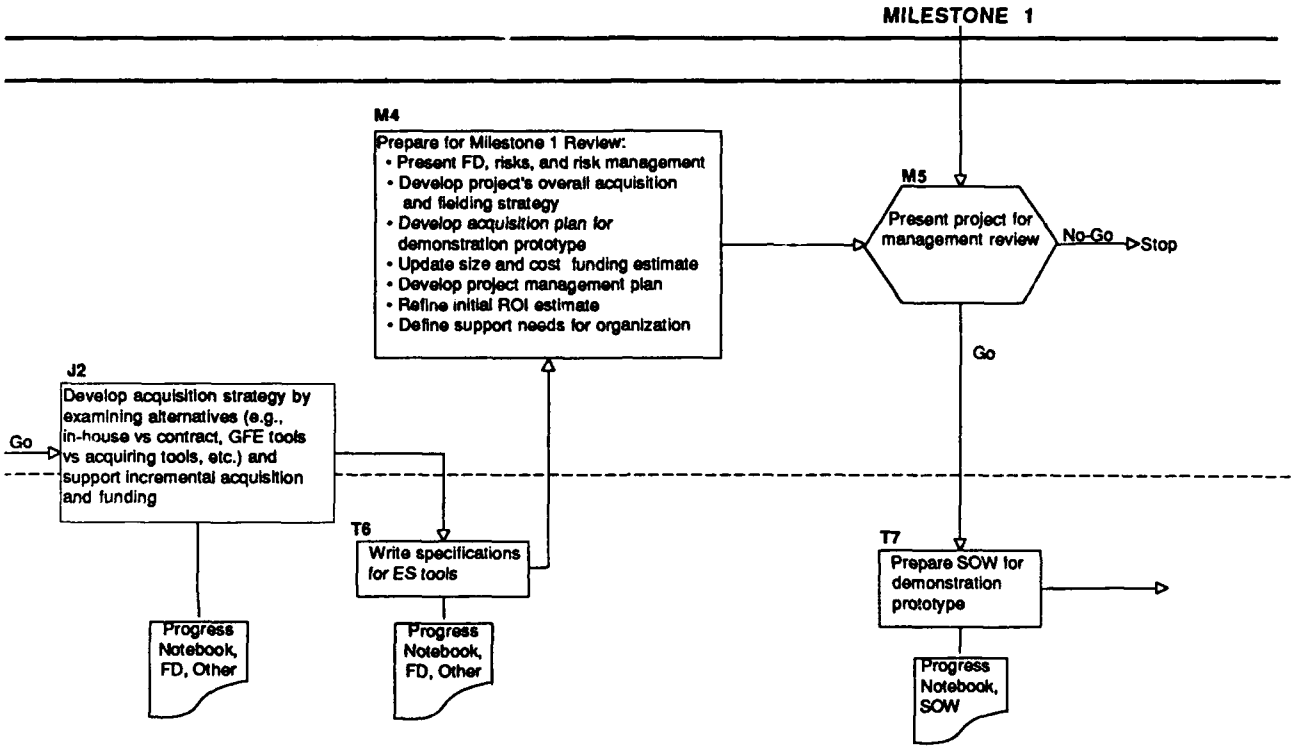


Figure 5.2—Concept Phase, Concept Prototypes


MILESTONE 1
M4
Prepare for Milestone 1 Review:
• Present FD, risks, and risk management
• Develop project's overall acquisition and fielding strategy
• Develop acquisition plan for demonstration prototype
• Update size and cost funding estimate
• Develop project management plan
• Refine initial ROI estimate
• Define support needs for organization
M5
Present project for management review
No-Go
Stop
Go
J2
Develop acquisition strategy by examining alternatives (e.g., in-house vs contract, GFE tools vs acquiring tools, etc.) and support incremental acquisition and funding
Go
T6
Write specifications for ES tools
T7
Prepare SOW for demonstration prototype
Progress Notebook, FD, Other
Progress Notebook, FD, Other
Progress Notebook, SOW

upon, a specification for the ES tool(s) is written. The technical process is documented in the Progress Notebook, and the technical results are used in preparing for the Milestone 1 Management Review. Following approval, a Statement Of Work (SOW) for the Definition/Design Phase Demonstration Prototype is written.

This chapter is divided into four sections: Section 5.1, this section, is an introduction; Section 5.2 presents an example risk containment table; Section 5.3 describes the management activities; and Section 5.4 describes the technical process.

5.2 RISK CONTAINMENT

Table 5.1 is an example prioritized list of major risk items that need to be identified and addressed during the Concept Phase. Since applications vary, the project manager should develop his/her own list of the top ten risk items.

5.3 MANAGEMENT ACTIVITIES

The management activities are organized into:

5.3.1 Initial Management Activities (M1)
5.3.2 Ongoing Management Concerns (M2)
5.3.3 Determining Life-Cycle Application Needs (M3)
5.3.4 Determining the Adequacy of the Functional Description (J1)
5.3.5 Developing an Acquisition Strategy (J2)
5.3.6 Preparing for the Milestone 1 Management Review (M4)
5.3.7 Holding the Milestone 1 Management Review (M5)

5.3.1 Initial Management Activities (M1)

The initial management activities are to: provide preliminary training for the expert, selected end-users, and others; provide tools needed to implement the conceptual prototypes; and to specify the documentation required during this phase.

Training for the domain expert should at a minimum consist of: an explanation of ES technology; provision of on-line experience in using an ES application; and help in developing a small (e.g., 10 rule) ES example. The training should not intimidate or overwhelm the expert (it should not be a crash course) and is best if it can proceed more or less at his/her speed. The training is of critical importance to the project effort. The better the expert understands the process, the more valuable he/she will be to the effort. At the very least, the expert will need to be able to understand the rules and knowledge structures used during the conceptual prototyping process.

Selected end-users who will be working with the KE to define the user interface should receive training. That training should include an explanation of ES technology and an on-line demonstration and hands-on use of the actual ES tool and workstation that will be used in developing the conceptual prototypes. This level of training should also be provided to other personnel from the maintenance, ADP, and security areas who will be participating in the Concept Phase effort.

The project manager (PM) should ensure that an adequate in-house ES tool is available to the KE for concept prototype development. If expert systems technology is not currently available in-house then the PM needs to establish such a capability. Additional tools that may be required include: a special graphics package and monitor for developing the user interface; and an audio recorder and audiovisual recording equipment for recording knowledge acquisition sessions, user interface sessions, demonstrations, and reviews. Other equipment may be needed to develop concept prototypes of ES/AIS integration or for

demonstrating the ES interface to a special device. The PM should try to anticipate these needs by working closely with the KE and the ADP support personnel.

Table 5.1

AN EXAMPLE PRIORITIZED LIST OF CONCEPT PROTOTYPE RISK ITEMS

	Risk Item	Risk Management Technique
1.	Personnel shortfalls	Staff this phase with top-talent analyst or KE; plan to acquire staff for next phase
2.	Lack of cooperation and participation across communities (developer, user, ADP, maintenance, security)	Explain need for team effort, ongoing community commitment; explain need for user community commitment and enthusiasm in particular; work with credible domain expert(s), end-users, and test cases; reach agreement with communities over life-cycle resource needs
3.	Straining of ES capabilities	Scope application to eliminate hard problems; demonstrate low-risk doable solutions in Concept Prototypes; avoid gold plating
4.	Difficulty integrating ES with AIS	Explore existing solutions; explore feasible low-risk designs; address security and telecommunication issues; get ADP and security support for developing and evaluating options
5.	Need for single-level or system high secure ES	Need to acquire staff with proper clearances (either contractor or in-house); be sure added costs of security do not exceed earlier cost benefit analysis (if so, project may be inappropriate at this time)
6.	Difficulty in developing acquisition strategy, incremental funding	Trade off cost effectiveness of design and development needs in specifying ES tools; trade off in-house vs contractual and variations; develop acquisition strategy that will easily support changes to plan, schedule, and funding
7.	Unrealistic schedule and expectations	Be realistic in plan and schedule; prepare user community for incremental acquisition/rapid prototyping approach requiring some flexibility in changing plan, schedule, and funding
8.	Documentation: maintaining the Progress Notebook; problems with documenting ES-specific elements	Designate a project librarian to be responsible for maintaining all documentation; develop plan for overseeing, reviewing, validating documentation beginning in next phase and continuing through deployment; specify how ES-specific elements should be documented
9.	Difficulty in doing cost benefit/ROI	Use conceptual prototyping experience as aid for improved cost benefit/ROI; compare with similar ES examples
10.	Validity of problem solution to mission	Determine that problem solution is still valid after FD changes through concept prototyping

The PM should define the documentation requirements for this phase and make them known to the development staff. The minimum amount of documentation (as shown in Figure 5.2) consists of the Progress Notebook—in particular, sections of the Progress Notebook that contain the functional description; the logbook kept by the KE during problem definition; and the Demonstration Prototype SOW. The PM should designate sections of the Progress Notebook for information, issues, and problems not provided for in the FD as described in DoD Standard 7935.1.

The FD begins in this phase (as shown in Figure 3.2) and will be developed on an ongoing basis. As the project unfolds, additional description is added to the FD. It becomes a history of problem description acquired during the Concept, Definition/Design, Development, and Deployment Phases. In the Concept Phase, the PM should designate a project librarian who will be responsible for maintaining the documentation. If the project effort is small, the PM or KE may have to double as the project librarian.

5.3.2 Ongoing Management Concerns (M2)

Ongoing management concerns include: carrying out the project plan; facilitating exchanges and cooperation among the user, ADP, maintenance, and security communities; monitoring feedback satisfaction and/or dissatisfaction from the user community; and soliciting project support.

The PM should continue to facilitate information exchange across user, developer, ADP, security, and maintenance communities by arranging meetings, explaining the need for a team approach to ES development, and offering seminars and training in ES. As the project becomes better defined and the PM is better able to estimate the level of effort required across the community, he/she should discuss these needs with the organizations and obtain assurances of the necessary support. The PM should frequently monitor the user community with regard to their satisfaction with the effort—the PM should not wait until the FD review.

As the Concept Phase progresses and the potential for a successful project increases, the project should be briefed to higher management in order to attract champions, generate enthusiasm, and ensure support. If possible, include in the briefing a videotape presentation of the conceptual prototype to illustrate how the problem will be solved. Having an enthusiastic member of the user community actively participate in the briefing will also be beneficial.

5.3.3 Determining Life-Cycle Application Needs (M3)

The life-cycle application needs to be addressed are categorized and described below:

5.3.3.1 Quality Assurance Guidance
5.3.3.2 Configuration Management Plans
5.3.3.3 Security and Integrity Needs
5.3.3.4 Test and Evaluation (T&E) Plan
5.3.3.5 Development Staffing Requirements
5.3.3.6 Roles and Needs of the User Community
5.3.3.7 Maintenance Needs and Plan
5.3.3.8 Training Plan
5.3.3.9 Telecommunication Needs
5.3.3.10 Documentation Needs

5.3.3.1 Quality Assurance Guidance

Quality assurance (QA) is an important consideration for any software application that requires a team effort and that will be maintainable and extensible—all qualities required by ES products.

The PM should (either alone or working with the KE):

- Establish naming conventions for ES elements such as objects, entities, attributes, rulesets, frames, procedures, etc. This is necessary to be able to share and reuse elements.
- Define an implementation style for each appropriate representation type, such as the allowable size and complexity of rules or frame or the definition of what constitutes a ruleset.
- Establish a formal style to be used for annotated descriptions of code and data. The formality should enable the automated use of the explanation, documentation, and other activities.

5.3.3.2 Configuration Management Plans

The PM should develop a configuration management plan for the next phases. All configuration management activities will be the responsibility of the project librarian for the Definition/Design, Development and Deployment Phases. All configuration management materials and support will be handed off to the maintenance organization at the completion of the Deployment Phase.

Many standards and tools have been established for software configuration management and are required in the development of large software engineering efforts. The methods used are good software engineering practice and should be utilized in ES development. Because ES tools have been built as single-user systems, many of them do not support configuration management; and even if they do, they do not support configuration management across multiple developers on different workstations. In the Sanders Associates, Inc., study [Bardawil 1987], several development projects (DEC/XCON, Lockheed/Expert Software Pricer, Lockheed-Georgia/Pilot's Associate) reported using the VAX/VMS Configuration Management System (CMS) for source file control. Others reported augmenting existing tools with their own code.

The PM should define the software configuration management methods to be used during the next three phases. This includes:

- Defining module, component, version, and release in terms of their attributes and defining the operations that will be performed by the configuration management system/librarian in managing them.
- Defining version control: Version control is especially critical for the coordination of knowledge base development among concurrent developers on separate workstations. An example of how version control might work is:
  - Developers will check out from the configuration manager/librarian a version of a module to work on, complete it, and return it to the configuration manager. The librarian can prevent multiple concurrent versions by checking out a module to a single user at a time.
  - The project librarian will oversee acceptance testing of the module and, after testing, release the module with a new version number and documentation.
  - Dependency graphs of module relationships will be supported so that changes in a module can signal changes to or retesting of larger components. Ultimately, one might want an automatic tool which would recompile source modules that had been changed and relink them into executable programs.
- Defining the problem structure in terms of the major components and modules (if possible at this time) that will be managed by the configuration system.

5.3.3.3 Security and Integrity Needs

Security and integrity needs are issues that require joint technical and managerial attention. Section 5.4.1.3 below discusses the technical role in understanding the security problem and representing the solution in a conceptual prototype. That prototype should be

reviewed by the security, ADP, and user communities to ensure that it meets needs and follows DoD security policy requirements.

After the technical need for security is recognized and the possible classification level(s) of the system are described, the PM will be responsible for consulting DoD security policy, documentation and personnel to determine the security level required for the development of the ES and for deployment of the ES product. The PM should ensure that security personnel are available to aid the technical staff in defining an initial architecture that meets the security requirements.

Protection against compromise or leakage of classified information becomes a requirement if any data used in the ES are classified or if the ES is interfaced to or embedded in AISs that are classified. In any case, all workstations, AISs and environments will need to conform to DoD security regulations governing computer system security. This will include physical security, personnel clearances, identification and authorization of users, correct classification markings on all paper and screen output, and the design, implementation, and maintenance of secure interfaces to any AIS. All AISs, ES workstations, and interfaces will require proper DoD accreditation. Maintenance personnel may also be required to have clearances.

Security may affect:

- The ability to acquire staff within a given period of time, particularly if they do not already have clearances.
- The definition/design and development environments with respect to cost of the physical layout and hardware and software selection.
- The deployment environment with respect to a secure system architecture that includes all AISs, ES workstations, and interfaces between system components.

The increased cost in building an ES application that satisfies security and integrity requirements is an important factor to consider in deciding whether or not to continue the project effort.

5.3.3.4 Test and Evaluation (T&E) Plan

The PM should develop a T&E plan that will cover T&E activities in the Definition/Design, Development, Deployment and Post-Deployment Phases. At the end of the Deployment Phase all T&E plans, software, and documentation will be turned over to the maintenance organization. This includes the documentation required by DoD Standard 7935.1: the Test Plan (PT) and Test Analysis Report (RT). Both documents will be part of the Progress Notebook. The PT documentation will begin in the Definition/Design Phase and the RT in the Development Phase in accordance with DoD Standard 7935.1. The plan developed now should be documented in some specific section of the Progress Notebook (perhaps marked "Initial T&E Plan"). Note that this is a draft T&E plan that will need refinement in later phases. Test and evaluation for post deployment will include an evaluation conducted approximately 6 months after the ES is in use to determine its benefit to the user organization, lessons learned, etc. This will be documented and shared with the community.

The designated test and evaluation person or team will address T&E throughout the life cycle. It is important that they test the user documentation as well as the software since the documentation is usually independently developed. The T&E leader must have some training in ES technology and some understanding of the domain and of integration issues. This person is critical to the success of the project. Cutting back on T&E activities increases technical risk considerably.

The information needed to develop a T&E plan includes:

- Problem related
  - Bounded problem description in terms of reasoning subcomponents and problem categories.

  - Definition of module, component, version, release.
  - Task breakdown, and list of identified modules and components (if possible).
  - Information about text and electronic knowledge sources.
  - Performance requirements.
- User related
  - Who, where, and how available the end-users and experts are.
  - Description of the deployment environment.
  - Whom to work with to acquire real test cases.
- Operation related: software configuration management methodology and placement of the T&E role within that activity.

Development of a T&E plan:

- Test cases: determine availability of adequate test cases statistically representative of the problem categories. "Adequate" means that their data needs are well specified and there are enough cases so that they can be divided into disjointed testing sets, each allocated to a different phase of development. The deployed system must be capable of handling all the cases. The plan should provide some cases in each set to the KE and development team to use in their development and testing efforts.
- End-user tests: develop plan for end-user participation in T&E and negotiate the plan with the user organization before writing SOWs.
  - Identify the number of users that will participate in testing at critical points in the Definition/Design Phase, the Development Phase, and the Deployment Phase. (These should not be the same end-users who are working with the development team.)
  - Work with the user organization to develop evaluation techniques to determine/measure the benefit of the ES to the user organization and changes to the way of doing business within the user organization as a result of using the ES. This evaluation could begin as early as the end of the Demonstration Prototype effort.
- T&E level: decide the granularity level at which T&E will be carried out (e.g., module, component, case).
- Acceptance testing
  - Lay out the T&E report format for acceptance testing at the end of the scheduled definition/design prototype iterations, development iterations, and deployment.
  - For Demonstration, Testbed, and Operational Prototypes: define plan for acceptance testing at each in-process review (IPR) and milestone to include:
    - Performance correctness.
    - Performance timeliness (where appropriate): timeliness measures and interpretation in terms of improvements to knowledge base organization or expected degradation as prototype grows in complexity.
    - Benefit to user organization.
  - During deployment
    - Interfaces to/from AIS(s): performance correctness and timeliness.
    - Special devices: performance correctness and timeliness.
  - ES
    - Internal performance correctness and timeliness.
    - System performance correctness and timeliness.
    - Benefit to user organization.
  - At end of deployment
    - Deliver documentation and relevant T&E methodology to the maintenance organization.

  - Post deployment
    - Evaluate benefit to user organization after elapsed time period (e.g., 6 months).

Critical factors to identify as a result of test and evaluation include:

- Degradation of timeliness with the size and complexity of the rulebase
  - Detection during prototyping may enable improvement through knowledge restructuring.
  - Problems due to garbage collection (e.g., reclaiming memory) may force consideration of a tool with developer control over garbage collection or a conventional implementation of the deployed ES.
  - Degradation during prototyping may indicate a need for improved heuristics, larger memory, faster hardware base, different tool.
- Architecture design to:
  - Meet availability requirement.
  - Include built-in ability to measure performance of external interfaces or, in case of embedded systems, internal interfaces.
- Recoverability of the system after different kinds of failures (needed for reliability and availability).

5.3.3.5 Development Staffing Requirements

The project manager working with the Knowledge Engineer should use the FD and technical experience gained from developing the Concept Prototypes to determine the number and skill structure of the staff necessary to develop the project.

Project roles in the development organization include:

- Project manager: controls and monitors the project and is responsible for motivating the project team; developing the project plan, schedule, reviews, and cost benefit/ROI; maintaining interactions across participating communities; defining QA methods, configuration management plan, and T&E plan; addressing security needs; addressing telecommunication needs; developing training plans; developing maintenance plans; etc.
- Project monitor: for any phase of development that is contracted out, there will be an internal government project monitor who will control, monitor, and review the project effort and aid the project manager in achieving his/her goals, especially with respect to user, maintenance, security, and ADP participation.
- Knowledge Engineer: analyzes the problem; specifies, designs, and evaluates the solution; works closely with the expert(s).
- ES programmer(s): designs details, implements the ES, and performs tests of ES units.
- Test and evaluation person/team: designs and runs tests, reviews progress, questions design and implementation, and applies acceptance criteria.
- Project librarian and staff: maintains documents (particularly the Progress Notebook), controls software configuration management, attends project meetings, and takes/collects the "official" project notes.
- Tool smith: maintains and enhances ES tools, dispenses tool expertise.
- Project statistician/administrator: collects and analyzes project data, which may include time/cost for specific tasks.
- ADP programmer(s): support interfaces to external databases, files, AISs, graphics, networks, special devices, etc.
- Security specialist: supports the staff (1) in developing a system architecture that is DoD accreditable and (2) in determining acceptable security environment and security functions required of the ES tool(s) during the pre-deployment phases.

The skill composition and size of the project team will depend to a large extent on the size and nature of the application. However, every ES project will require a project manager, KE, and T&E person. In addition, an ADP programmer or system specialist is required if AIS integration or a special device interface is needed; and a security specialist is required if the ES will be deployed as a secure system. The duties of project librarian, project statistician/administrator, and tool smith may be filled by other project members if the project is a small effort.

5.3.3.6 Roles and Needs of the User Community

The PM has to be knowledgeable about the user organization and needs to ensure that the ES product will fulfill the users' mission needs and expectations. Information necessary to do so is discussed here. The actual collection of this information should be done under the guidance of the PM but may be carried out by the KE or other technical staff as part of the knowledge acquisition activity.

User community roles of interest to the PM are:

- End-users: will integrate the ES product into their daily work environment.
- Expert(s): may or may not be an end-user; is articulate; is a recognized, authorized expert in the problem domain; and communicates knowledge about the problem domain and solution while working with the KE to build the ES.
- Manager: manages end-user organization; is responsible for assuring resources and availability of end-users and expert(s); participates in project reviews; and supports the project effort.
- Problem owner: may be any of the above; he/she is responsible for recognizing the problem and seeking a solution, participates in project reviews and supports the project effort. Information is required about experts and users in order to plan their participation during the Definition/Design, Development, and Deployment Phases. Deployment information is needed in order to specify ES tool requirements and to size and cost the effort.

The following information should be gathered, documented in the Progress Notebook, and kept up to date with direction and support from the project librarian:

- Identify the users and describe their job tasks in relation to the problem. Be explicit, particularly if the problem space (defined in Section 5.4.1.1) partitions into tasks addressed by different groups of users that include specialized experts.
- Identify the number and location of users and experts now and expected in the Post-Deployment Phase.
- Describe the equipment being used now and expected to be used in the Deployment and Post-Deployment Phases.
- Describe the job turnover rate for users and experts.
- Describe the availability of the users and experts during the different phases.
- Describe the credibility and authority of all experts who will work with the KE.
- Describe current user and expert skill levels with respect to the domain, computer usage, and expert systems.
- Discuss user needs for ES training.

The following information about managers and the problem owner should be collected, entered in the Progress Notebook, and kept up to date:

- Role in the organization.
- Location.
- Skill levels with respect to the domain, computer usage, ESs, and software development projects.

5.3.3.7 Maintenance Needs and Plan

Maintenance issues related to management are addressed here. Maintenance issues related to technology are addressed in Section 5.4.2 and cover an estimate of the maintenance needs for the next phase, the role of the Knowledge Base Administrator, and a discussion of how the levels of maintenance can be simplified by the right choice of ES tool and workstation.

The PM has to be knowledgeable about the maintenance organization that will be maintaining the deployed ES. It is important that he/she establish contact with a liaison person in the maintenance organization to work with him/her and the technical staff on maintenance and deployment-related concerns. This includes identifying the skills needed to maintain the deployed ES that are not currently present in the maintenance organization. Since a lack of these skills poses a potential risk, the PM should consider ES architecture alternatives that would reduce the risk. The information gathered about the maintenance organization, skills needed, alternative ES deployment architectures, and an initial maintenance plan should be documented in the Progress Notebook. Recommendations about the characteristics of the deployed ES tool will be used in developing ES tool specifications.

Information needed about the maintenance organization includes (1) the location of maintenance people with respect to user community plans for ES deployment and (2) the expertise level of maintenance people with respect to specific hardware/software, computer languages, and tools.

The roles that maintenance personnel may need to fill to support ES technology are described below:

- Knowledge Base Administrator (KBA): maintains knowledge base; tests, evaluates, and releases system whenever changes occur to the knowledge base or to underlying software/hardware; and has responsibility for managing the configuration of the deployed ES environment including external interfaces. The KBA also has responsibility for maintaining computer-aided instruction (CAI) material, training end-users, disseminating documents, and training for new releases.
- Tool smith: maintains ES tool, installs new versions, dispenses tool expertise.
- Workstation maintenance person (may be in-house, contractual, or vendor): includes hardware/software (HW/SW) maintenance of special-purpose AI workstations (e.g., Symbolics, TI Explorer), engineering workstations (e.g., SUN Microsystems, Apollo), or standard workstations (IBM, Apple Macintosh), software includes operating system and off-the-shelf packages such as spreadsheet and DBMS.
- ADP programmer: maintains AIS/ES interfaces and changes to AIS, networks, and I/O devices.

Depending on the ES architecture, maintenance may be required at four levels:

Maintenance Level	Personnel
Level 4: ES	KBA
Level 3: ES tool	Tool smith
Level 2: Workstation HW/SW	Workstation maintenance person
Level 1: Integration	ADP programmer

The maintenance personnel and their tasks may not be currently supported, except for ADP and standard workstation maintenance personnel and their tasks. Advanced workstations and/or ES tools may not be currently owned or maintained by the government and, of course, a KBA will not be necessary until the ES becomes a product. Note that although the KBA is shown as part of the maintenance team, he/she will most likely reside in the user organization. With so many levels of maintenance, it is obvious that personnel at

different levels will need to be closely coordinated. Plans to make changes at lower levels will require concurrence with maintainers at higher levels. Possible ways to simplify the levels of maintenance are discussed in Section 5.4.2. Briefly, they require using common workstations and employing an ES that is implemented in a conventional programming language.

The PM should prepare a preliminary maintenance plan for deployment and post deployment based on user plans and needs and the recommended ES architecture with respect to workstation and language. The plan should identify the anticipated post-deployment user sites and number of ES workstations to be supported, the number and location of maintenance personnel needed to support each site, and the role and responsibilities of the KBA(s). In addition, the plan should define all documentation (as described in DoD Standard 7935.1) and software to be handed over to the maintenance organization at the end of the deployment phase.

5.3.3.8 Training Plan

The PM needs to define the levels of training required for participating personnel in all communities (developers, users, maintainers, ADP, security) and to develop a training plan.

Table 5.2 shows an estimate of the level (low, medium, high) of information needed by various personnel. A definition of these levels by information area follows.

- Low: a general introduction to AI and ES that includes system fallibility in terms of "garbage input produces garbage output". Since the tools lack the capability to check rule consistency, end-users may inadvertently introduce errors if they add

Table 5.2

LEVEL OF INFORMATION NEEDED BY PROJECT STAFF

Role	ES Training	Domain Understanding	Integration Understanding
Project personnel			
Project manager	Medium	Medium	Medium
Project monitor	Medium	Medium	Medium
Knowledge Engineer	High	High	High
AI programmer	High	High	*
T&E leader	Medium	Medium	Medium
Librarian	Medium	Medium	Low
Tool smith	High	Low	Low
Project stat/admin	Low	Low	Low
ADP programmer	Low-medium	Low	High
User personnel			
End-user	Low-medium	*	Low
Expert	Low-medium	High	Low
Manager	Low-medium	Medium-High	Medium
Problem owner	Low	High	Low-medium
Maintenance personnel			
KBA	High	High	Medium
Tool smith	High	Low	Low
Workstation maintenance	*	Low	Low
ADP programmer	Low-medium	Low	High

*Application dependent.

their own rules—even the experts make mistakes. They should be taught not to believe everything an ES tells them. In addition, there should be hands-on experience using an ES to solve a problem or two. Estimated training length is 2–3 weeks for end-users without any computer experience, 1 week for ADP personnel.

- Medium: equivalent of "low" training plus course material on ES paradigms or representation models and on using the programming environment, followed by hands-on experience building a small knowledge-based system (KBS). Length of course should be about 3 months for ADP personnel.
- High: equivalent to college training in understanding technology; using LISP, PROLOG, and C; and experience in building tools or in using tools to build ESs. Includes understanding problem-solving methods, and representation models, mapping representation models to tool implementation, and using the tool programming environment.

Domain understanding training for low, medium, and high levels:

- Low: information gathered by selected readings, briefings, and seminars over a 1–2 week period.
- Medium: equivalent to novice end-user.
- High: equivalent to intermediate end-user.

System-integration understanding needed for low, medium, and high levels:

- Low: understanding the integration problem at a system level.
- Medium: understanding the problem deeply enough to be able to participate in the evaluation of solutions presented by those with a high level of understanding.
- High: understanding the problem in great detail including design, implementation, testing steps that need to be taken to build the interface, amount of effort, time frame, etc.

As shown in Table 5.2, different types of people will require varying degrees of training.

- End-users: will require training in using the ES and the workstation environment. We may like to think that the ES will provide built-in training, but this will probably not be the case. Formal computer-aided instruction (CAI) materials and a training course may need to be developed, especially if there are a large number of users and a high turnover rate. The KBA should be responsible for maintaining the CAI materials and training after deployment. Training will need to be carried out for different end-users at different phases during project development.
- User organization personnel: the experts, problem owner, and project manager should receive at least low-level ES technology training early in the Concept Phase. Once it has been decided to continue with the effort, they should receive the appropriate level of training. End-users that will participate in knowledge acquisition or T&E should receive training immediately prior to their participation. General end-user training in using the deployed system should be scheduled during the deployment phase. Deployment will most likely be phased (if there are many users) and the training will need to be coordinated with the deployment schedule.
- Maintenance personnel: the liaison maintenance person should receive training in ES technology and the problem domain during the Concept Phase in order to provide information about the ability of the maintenance organization to provide the capabilities needed.
- Project personnel: should receive training in ES, problem domain, and integration during the Concept and Definition/Design Phases. Some of the integration information may be acquired later on if the AISs undergo change. In fact, the ADP programmer may be a periodic participant in the project, being called upon to offer integration updates and plans at critical decision points.

In addition, certain project personnel require specialized experience and/or skills, which may be available in short courses or seminars or may have to be learned by experience on the project.

- Project manager/monitor: needs training in project management and development of application products using ES technology.
- Knowledge Engineer: needs experience/training in interviewing techniques and the knowledge acquisition process.
- T&E leader: needs experience/training in T&E, training in development of ES evaluation criteria, and training in conducting ES tests and reviews (possibly available in professional seminars).
- Librarian: needs experience/training in maintaining documents and in software configuration management, needs training in ES software configuration strategy (possibly available in professional seminars), and needs good organization and writing skills.

5.3.3.9 Telecommunication Needs

The PM should determine the extent of the telecommunication needs required by the deployed ES. This will require knowledge about the dispersal of the deployed system, what AISs it will be interfacing to, the estimates of the user load by time of day, and the estimated amount of data transaction and processing that will need to be supported. This estimate, if large, could be a decisive factor in considering alternative architectures.

For example, if the ES is to be deployed worldwide to thousands of users, all of whom need to access a centralized database, it may be possible to distribute copies of the database or snapshots of often-used data in a timely fashion to local data servers. This would reduce wide area network (WAN) traffic considerably but would probably require several/many new local area nets (LANs) to connect local users to their data server. Such a solution may not be feasible if the data change rapidly (and the users require timely data), the pattern of data needed is not easily determined, and the total database is very large (the frequent moving of large databases may put a big strain on the WAN). Cost and functionality trade-offs may need to be done that extend beyond the current project in that the needs of other users may have to be assessed and satisfied.

The analysis of telecommunication needs should be documented in the Progress Notebook, and if the needs are excessive, they should be presented as a risk factor for further investigation. Alternative architectural solutions may be presented as ways to resolve the risk and should include estimated cost/benefit analyses.

5.3.3.10 Documentation Needs

The PM should lay out the documentation requirements for the entire project by phase. These should satisfy the requirements shown in Figure 3.2 taken from DoD Standard 7935.1 but may include additional documents.

In particular, the PM should specify the sections in the Progress Notebook for information collected before it is required by DoD Standard 7935.1 (e.g., the T&E plan in this phase). As an example, the PM could require that the initial T&E plan be documented under the PT (though the PT is not required until the next phase), or the PM may name a new section of the Progress Notebook, say, "Preliminary PT" to contain this information.

The PM should specify how ES knowledge should be documented; for example, the problem in documenting objects and frames was discussed in Chapter 3. Most of the representational knowledge structures used in ESs can be considered as data and process. We do not propose to split them apart in order to document them as conventionally required in the FD and SS under data and functions. Rather we recommend that the PM decide which category they should be documented under, note this early in the document, and state the decision again in both the data and function sections of the document.

Documentation requirements should be recorded in the Progress Notebook.

5.3.4 Determining the Adequacy of the Functional Description (J1)

Determining the adequacy of the functional description is a joint technical and management activity. It is addressed in Section 5.4.4 as part of the technical process.

5.3.5 Developing an Acquisition Strategy (J2)

Development of an acquisition strategy that effectively integrates the technical, business, and management elements of the project is a joint technical and management effort. It is discussed in Section 5.4.5 as part of the technical process.

5.3.6 Preparing for the Milestone 1 Management Review (M4)

The main purpose of the Milestone 1 Management Review is to show that we have a reasonable definition of the problem and a reasonable low-risk plan for solving it. The outcome of this review will be to decide whether to continue the project.

Information to be presented at the review includes:

- A discussion of the FD as a result of the conceptual prototyping effort, including alternative architecture possibilities and analysis and management.
- A validation of the need of the user community for the problem solution.
- The acquisition and fielding strategy for the project.
- The acquisition plan for the Demonstration Prototype effort.
- Update of the rough size and cost estimate presented at Milestone 0.
- A project management plan and schedule.
- Refinement of the cost benefit/ROI presented at Milestone 0.
- Description of the project support needs.

Functional Description. A high-level walk through the functional description should be prepared. It should discuss the problem, alternative architectures, and recommended solution. A short video presentation of the concept prototype may be an effective way to demonstrate how part of the problem can be solved. Specific risks to be addressed are those that strain ES capabilities, difficulties in integrating the ES with AISs or special devices (if appropriate), the impact of security on the architecture (if appropriate), and the impact of telecommunication requirements on the architecture (if appropriate).

The combined impact of integration, security, and communication requirements on an ES application can be substantial. If such is the case, alternative architectures should be presented along with a recommended architecture or a plan for conducting a more in-depth alternative architecture study in the next phase. If there is doubt that a cost-effective architecture can be defined or reason to believe that the architecture decision could invalidate the Demonstration Prototype, then the prototype effort should be delayed until the architecture has been decided.

Validation of Need. Since the concept prototyping may have resulted in narrowing the problem scope, it is important to show that the problem as currently defined is still important to the mission of the user community and is supported by that community.

Acquisition and Fielding Strategy. The acquisition and fielding strategy discussion should include the acquisition alternatives considered: ES tool specifications, contracting vs in-house, and the type of procurement and its effectiveness at satisfying the incremental development and funding needs of ESs. Plans for acquisition, staffing, training, and deployment (including maintenance considerations) during the life-cycle phases should be presented.

Acquisition Plan for the Demonstration Prototype Effort. This plan should be presented in detail. It should include a description of the requirements that will be written into the SOW to control the risk (e.g., ES tool specifications, security requirements, contractor experience). It should also describe how the PM intends to acquire the Demonstration Prototype effort and furnish the resources needed, such as staff, workstations, and tools.

Size and Cost Estimate Update. The results of the problem exploration and life-cycle considerations pursued during this phase should provide an improved cost estimate over that presented at Milestone 0. The cost estimate can be based on:

- The logbook kept by the Knowledge Engineer during development of the Concept Prototypes, which provides an estimate of the amount of effort required to develop the first case example. This could be extrapolated for other categories of the problem to estimate a level of effort necessary to accomplish knowledge acquisition/ES development for the Demonstration Prototype. This would take into account the effort of the KE, AI programmers, expert(s), and end-users. Extrapolations can be made from the Demonstration Prototype to Testbed and Operational Prototypes, but these will obviously be very rough.
- Estimate of the size of electronic and text source data/knowledge, and an estimate of the effort needed to acquire it for the stand-alone prototype efforts and for the operational prototype and deployed system.
- System architecture costs (integration with AIS(s) and special devices, security, communications) estimated from the Concept Prototypes developed during this phase.
- Estimated cost of: maintenance during development, deployment, and post deployment (including the KBA maintaining the ES and changing knowledge base); training of development staff users, experts, and maintenance personnel; T&E; configuration management; and documentation.
- Estimated cost of project management activities and reviews. The incremental development/rapid prototyping approach will result in many changes to schedules, plans, and costs. Therefore changes need to be handled in a cost-effective manner as an integral part of the development process.
- As in the Milestone 0 cost estimate, costs of other similar efforts should be compared with the new cost estimate to ensure that it is not off by an order of magnitude or more.

Project Management Plan and Schedule. An estimated plan and schedule consistent with the cost estimate should be developed for the project life cycle. The plan and schedule will be firm through the first iteration of the Demonstration Prototype, less precise for the rest of that development, and increasingly less precise for each successive prototyping phase.

Cost Benefit/ROI Refinement. The PM should prepare a refinement of the cost benefit/ROI presented at Milestone 0. It should be based on changes in the problem definition during the Concept Phase. Examples of such changes that may possibly affect cost benefit/ROI are: a narrowing of the problem scope, which may reduce the benefit; additional functionality, which could increase the benefit; and changes in architectural designs required for integration, security, and telecommunications, which could substantially increase the cost of the solution.

Project Support Needs. Describe identified project support needs for the Definition/Design, Development, Deployment, and Post-Deployment Phases from: the user community, the maintenance community, the ADP organization, and the security organization.

5.3.7 Holding the Milestone 1 Management Review (M5)

The last management activity in the Concept Phase is to present the project at the Milestone 1 Management Review. We strongly recommend that managers at all levels of the participating organizations be encouraged to attend, as well as expert and end-user representatives from the user organization. As mentioned earlier, a videotape of the prototyping effort may be an effective way of describing a part of the problem and its solution.

5.4 TECHNICAL PROCESS

The technical process in the Concept Phase consists of seven tasks, each of which is addressed in a separate section below.

5.4.1 Developing the Concept Prototype(s) (T1–T3)
5.4.2 Defining Other Application Characteristics (T4)
5.4.3 Developing the Application Checklist (T5)
5.4.4 Holding the Review to Determine FD Adequacy (J1)
5.4.5 Developing an Acquisition Strategy (J2)
5.4.6 Writing the ES Tool Specifications (T6)
5.4.7 Writing the SOW for the Demonstration Prototype (T7)

Developing the concept prototype(s) further explores the problem by using in-house ES and other support tools to develop rapid prototypes of parts of the problem solution, especially those areas identified as risk items in the Concept Phase. The result of this activity is not a single integrated concept prototype but rather many small prototypes. These may demonstrate: how the general problem solution will work, how parts of the problem solution that strain ES capabilities might be handled, how the user interface may be structured, how integration with an AIS and/or special devices will be handled, and how security may be handled. Exploring the problem through rapid prototyping requires a team effort of developers, users, and ADP staff and the availability of in-house tools rich in functionality and flexibility.

Concept Prototyping is the first of four prototyping stages in the development of an ES using a risk management approach. All stages require a team support effort—particularly the KE and development staff working closely with the domain expert(s) and end-users. The three prototype stages following the Concept Prototypes include the Demonstration Prototype, the Testbed Prototype, and the Operational Prototype, which were introduced in Chapter 3 and will be described in greater detail in Chapters 6, 7, and 8.

The task of defining other application characteristics addresses maintenance and performance. It considers the technical issues of maintenance in the next phase and of maintaining the ES product, particularly with respect to the role played by the Knowledge Base Administrator (KBA). These technical considerations are input to the PM, who will be planning and estimating the cost of the maintenance program. The performance requirements for the application are derived in cooperation with the user community and use information and experience gained from developing the Concept Prototypes.

The application checklist is developed from the results of those technical and management activities that precede it (see Figure 5.2) which have addressed quality assurance, T&E, configuration management, maintenance, and training. Weights and priorities are assigned to characteristics by development phase in order to derive a profile for each phase. These profiles are important aids in developing specifications for tool evaluation and selection, in developing the acquisition strategy, and in writing the SOW.

Determining the adequacy of the functional description is a joint effort between the technical staff and the PM. Its purpose is to be sure that the description is adequate to develop an acquisition strategy. Developing an acquisition strategy is a joint effort between the technical staff and the PM. Possible alternatives to be considered include whether the program should be implemented in-house or contracted out; how ES tools are to be acquired (e.g., from an approved government list, as Government Furnished Equipment (GFE), purchased by the contractor); and what (if any) are the language and/or ES tool restrictions imposed by DoD policy on the deployed ES product.

After the acquisition strategy has been decided, specifications for the ES tools are written. These will be presented at the Milestone 1 Management Review and used in writing the SOW.

The last task of writing the SOW is performed after the Milestone 1 Management Review has determined that the project should be continued. The SOW may be restricted to development of the Demonstration Prototype or may include later phases.

5.4.1 Developing the Concept Prototypes (T1-T3)

The process of further defining and understanding the problem and the solution is a team effort carried out by at least the KE(s), analysts, domain expert(s), end-users, and ADP specialist. Personnel with specific expertise in security, communications, special devices, etc., may be consulted and/or participate in the effort on an as-needed basis.

The conceptual prototyping process can be split into three tasks:

5.4.1.1 Defining the Problem (T1)
5.4.1.2 Exploring Integration with AIS and Special Devices (T2)
5.4.1.3 Exploring Security and Integrity Needs (T3)

First the problem, user interface, and knowledge sources are explored and better defined. Next, integration needs are explored and better defined using the problem definition. Last, the technical security and integrity needs are explored and better defined with particular attention given to how meeting the needs can affect the integration and architecture.

5.4.1.1 Defining the Problem (T1)

This task consists of:

- Exploring the problem space.
- Addressing risks in straining ES technology.
- Exploring programming and representation methods.
- Exploring the user interface.
- Defining the knowledge sources.
- Documenting the Progress Notebook.

Exploring the Problem Space. The process of exploring the problem space (defined below in this section) and illustrating how the problem is understood through conceptual prototyping is dependent on the KE(s) acquiring knowledge about the problem from the expert(s), end-users, written materials, courses, etc. Knowledge acquisition is recognized as the biggest bottleneck in building ESs. Though system analysis and interviewing techniques are used, the successful capture of knowledge is not a well-understood process.

The knowledge acquisition process requires the close interaction of the KE and expert and, to a lesser degree, end-users. Concept prototyping is an iterative process. Knowledge acquired by the KE from the expert and other sources is digested and implemented into a rapid prototype that is reviewed/used/critiqued by the expert to identify misunderstandings, errors, shortcomings, and insights, which are communicated to the KE for the next round.

Several equipment aids may be used in capturing knowledge: an audio recorder, an audiovisual recorder, and an ES tool. Schoen and Sykes [Schoen 1987] strongly urge the use of audiovisual techniques for all knowledge capturing sessions. For practical reasons, this may be too expensive. However, if the problem solution is visual and auditory (e.g., several people interact to solve it), then it would be beneficial to tape some problem-solving situations for later reference. This technique may also be used for selective sessions of high potential value (e.g., exploring the user interface with end-users or reviewing rules with other experts). Of most importance is the use of an in-house general-purpose ES tool that incorporates many programming and representational methods for the KE to use in exploring the problem space and evaluating ES techniques.

The conceptual prototyping process should be documented in a logbook, either in paper or electronic form, filled out by the KE to describe each day's activities. This logbook is an

important component of the Progress Notebook detailing the refinement of the conceptual prototype. In particular it should report:

- Meetings with the domain expert
  - Areas discussed and work accomplished in terms of problem cases and steps
  - Any new issues or concerns
  - Plan of action for next meeting
  - Tape identification if recording device was used
- Meetings or telephone discussions with other experts, users, developers of similar ESs, managers, etc.
  - Identity of person(s)
  - Subject
  - Information gathered
  - Tape identification if recording device was used
- Solitary work
  - Identification of: categories, reasoning subcomponents, and cases of current problem
  - Identification of activity: e.g., writing rules, reorganizing rules/knowledge, trying out different representations, user interface development, analyzing elicitation session

Information from the logbook will be useful later in developing estimates of amount of effort, size of the knowledge base, and cost for the prototype development.

The term, *problem space*, refers to a two-dimensional view of the problem where the number of problem *categories*, each made up of *cases*, defines the breadth, and the level of detail in the *reasoning subcomponents*, the depth. This is illustrated in Figure 5.3. The term *category* refers to a well-defined class or set of situations within the problem which are solved in a similar manner. The term *case* refers to a complete problem instance in a particular problem category. The term *reasoning subcomponent* refers to a collection of heuristics that support a specific analytic step. The solution thread for a given case is an ordered collection of reasoning subcomponents that have been dynamically specialized to address the problem at hand. The *scope* of the problem defines the bounds of the solution in terms of categories and reasoning subcomponents.

In examining the appropriateness of the problem to a KBS solution in the Initiation Phase, we determined that a useful part of the problem was capable of being solved. Now, we want to explore that problem space more deeply and systematically to determine: those parts of the problem that can be feasibly solved using existing and proven ES and computer science techniques; those parts of the problem still dependent on stretching ES technology; the management of these risks; and the rough order in which the problem solution can be incrementally implemented in terms of categories and reasoning subcomponents.

An example of how to explore the problem space for a problem with fairly well-defined categories and steps is given below.

The KE and expert need to work together in the following order:

- Step 1: Explore the *depth* of the problem by walking through typical cases for each category and decomposing them into reasoning subcomponents.
- Step 2: Have the expert describe each reasoning subcomponent in terms of its:
  - Overall goal.
  - Functions and decision processes.
  - Knowledge and data requirements.
  - Performance constraints.
  - User interactions.

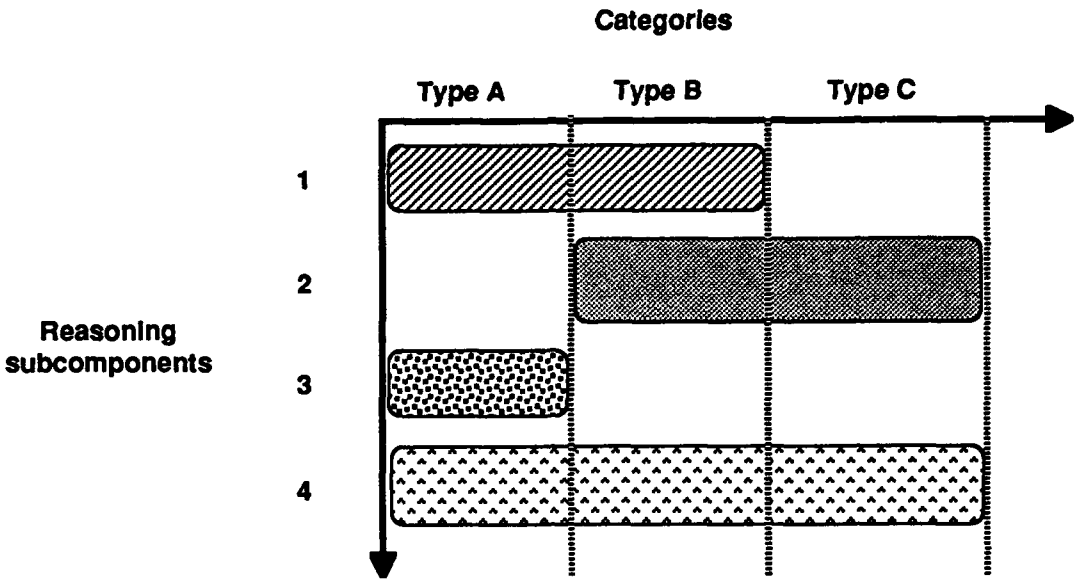


Figure 5.3—The problem space

- Step 3: Explore the *breadth* of the problem by defining the set of non-overlapping categories that describe that breadth and then developing an algorithm to rank the categories. An example of such an algorithm would be the following:
  - Select and assign a weight to each category criterion (e.g., degree of difficulty, frequency of examples, importance in terms of expertise needed, similarity to other categories).
  - Order the categories along each criterion dimension from least to most important.
  - Compute the combined "value" of each category by summing the products of the rank orders of the categories times their weights for all criteria.
  - Select the highest ranking category and fully explore a typical case of it. If this is unsuccessful, take the next highest ranking category and fully explore an example of it and so on. If no category can be successfully decomposed into ordered collections of reasoning subcomponents, then define a shallower (but still meaningful) problem space and start again by exploring the highest ranking problem category.
- Step 4: When a typical case has been successfully explored, use its category as the center of a "difficulty" continuum upon which the other categories will be located.
- Step 5: Select categories at intervals on the continuum, walk through a typical case from each of those categories, and estimate the relative ease/difficulty of implementing it with respect to the center category. This will become input to bounding the prototype effort and estimating the size of the effort.

The strategy to be used in the in-depth exploration of a typical case for a given category is for the KE to do the following:

- Step 1: Ask the expert detailed questions about each step of the solution.
  - Reconfirm the decision being made at that step.
  - Identify the data and knowledge used in making the decision and explain how they were used. If the expert cannot do this, use special elicitation techniques (described below).
  - Identify any performance constraints, for example:

    - Only one best decision/choice is possible;
    - Any decision/choice that fulfills certain requirements is acceptable;
    - All decisions/choices should be pursued or presented to the user;
    - Decisions/choices should be made with/without time constraint considerations.
  - Identify any required interactions with the user:
    - Describe best-judged presentation technique.
    - Describe kinds of explanation and knowledge browsing that could result.
  - If any step is too large, ask the expert to break it down into two or more steps and repeat from step 1 above.
- Step 2: Analyze the information collected; ask the expert questions as needed.
- Step 3: Review the KE's analysis with the expert.
- Step 4: Implement the case in a knowledge base (user interaction can be very crude for now).
- Step 5: Have the expert solve the same case using the expert system.
- Step 6: If it does not work, then investigate why, and fix it.
- Step 7: If it works but the expert is not happy with the way it works, ask for specific recommendations to make it better.
- Step 8: Have the expert review the rule set and comment on it. Most likely he/she will find it too specific and may offer more general rules. If so, rewrite with his/her suggestions and repeat from step 4.

Knowledge elicitation: if the expert cannot identify the data or knowledge used or explain how it was used, the KE should try to elicit insight by presenting hypothetical cases that he/she generates by varying the data/knowledge in a controlled fashion. The KE should ask the expert what his/her decision would be in those cases. If the decisions vary, then the pattern of variance may give the expert insight into a theory about how he/she is using the information. If not, or if this is not helpful, then in step 3 the KE should implement the rule explicitly. If it is not possible to represent the knowledge, the KE should consider the step as insolvable.

Addressing Risks in Straining ES Technology. Any part of the problem requiring an ES capability that is not well proven or supported (still in the research stage of development) must be explored in detail and a recommended solution or alternative solutions documented.

For example, if the expert and users expect that ES rules will be changed often (perhaps even locally by end-users), then support for rule consistency may be an important part of the problem solution. ES support for rule consistency is a research problem and may not be commercially available or proven in time for the production ES. The KE may try to work around this problem by exploring the possibility of partitioning the rules into small rulesets and organizing the rulesets into a structure that would be intuitively easy for the expert and users to browse. A new rule would be added, an old rule changed, or an old rule deleted by having the expert or user browse the rulesets and identify the ruleset to be modified. After the rule is modified, it would be the expert or user's responsibility to ascertain the consistency of the resulting ruleset. An important tool for exploring the consistency of a changed ruleset is a large test suite with known correct outputs. If the KE found that the rules were not easily organized into rulesets, then he/she must identify this as a high-risk issue and, if it is very important to the success of the ES, recommend that the project be discontinued until the capability is available or a different, low-risk means of solving the problem is found.

Exploring Programming and Representational Methods. The value of using different programming and representation methods should be explored by building small feasibility prototypes. For instance, a part of the case example that was implemented using only IF-THEN rules may be compared with an implementation using objects or with one using a combination of objects and rules. The goal is to identify the best method(s) to use in terms of

making the implementation easier and more natural for each reasoning subcomponent. Programming and representational methods are discussed in Appendix A.

Exploring the User Interface. After the KE and the expert have worked together to build a small concept prototype of the problem thread, they need to work with selected end-users to identify the I/O features most suitable for the user interface. The types of I/O presentation (e.g., forms, graphics, interface devices, text, windows) should be described and demonstrated. The users should indicate their preference for such things as: organizing and managing windows, how browsing should be conducted, the kinds of graphics needed, how they might like to access other tools such as a spreadsheet or editor, whether distinctly different interfaces are needed for different kinds of users (e.g., naive and advanced), and the ways in which they would like "explanation" and "help" to work.

Small user-interface Concept Prototypes should be built for, demonstrated to, and used by the end-users to elicit feedback that will result in improvements to the prototype. The result should be a definition of the initial functional requirements for the user interface. These prototypes may be implemented using the ES tool or, if the ES tool is lacking in functionality (such as advanced graphics or window handling), may require use of other tools.

Defining the Knowledge Sources. While exploring the problem space, the expert will identify and describe the knowledge sources used in developing the problem solution. Though the knowledge sources were identified and estimated in the Initiation Phase, we now need to probe deeper and be more explicit about their characteristics in order to estimate the effort and cost of acquiring and using them in later development phases.

The descriptions developed here should be entered into the functional description section of the Progress Notebook. They should be re-examined at various points in the life cycle and modified as necessary, and the resulting effect of any modifications should be reflected in overall size and cost estimates.

Information for text sources includes:

- Document name, source of document, and classification level
- For Demonstration and Testbed Prototypes the following information is needed for each type of data or knowledge:
  - Data or knowledge description
  - Why and where the data/knowledge is needed by the ES
  - Identification of contact person
  - Document page numbers
  - Realistic size, e.g., bytes, rules
  - Plans and costs for data acquisition
  - Based on examples, plans, and costs for knowledge acquisition (for use in prototyping)
  - Plans for handling updates to data values and to knowledge
  - Plans for handling updates to format changes
- For Operational Prototype and deployed system the following information is needed for each type of data or knowledge:
  - Data or knowledge description
  - Why and where the data/knowledge is needed by the ES
  - Identification of contact person
  - Document page numbers
  - Realistic size, e.g., bytes, rules
  - Plans and costs for data acquisition
  - Based on examples, plans, and costs for knowledge acquisition (for use in deployed system)
  - Plans for handling updates to data values and to knowledge
  - Plans for handling updates to format changes

Information for electronic data / information includes:

- Database/filename that contains the data/information, or application name if the data will be furnished by an application
- Classification level
- Current AIS managing database, file, application
  - Hardware: hardware base, storage media, network connections, other
  - Standard software: operating system, DBMS, file system, spreadsheet, other
  - Nonstandard software: "home grown" AIS, application, customized interface, other
- Future AIS (if planned) that will manage database, file, application
  - Development and installation schedule
  - Hardware: hardware base, storage media, network connections, other
  - Standard software: operating system, DBMS, file system, spreadsheet, other
  - Non-standard software: "home grown" AIS, application, customized interface, other
- Database or file or application description
  - Listing or document of data schemata or format information
  - Description of data elements needed by ES and why and where they are needed
- For Demonstration and Testbed Prototypes
  - Realistic size of data/information
  - Plans and costs for acquisition data
  - Identification of contact person
  - Plans for handling updates to data/information values
  - Plans for handling updates to format changes
- For Operational Prototype and deployed system
  - Estimated size (e.g., bytes, rules)
  - Plans and estimated costs for data/information acquisition
  - Plans for identifying contact person
  - Plans for handling updates to data/information values
  - Plans for handling updates to format changes

Documenting the Progress Notebook. After the problem has been explored in depth and breadth and the initial user interface defined, the Progress Notebook will be updated to reflect the refined problem description. As shown in Figure 3.2, the functional description document begins in the Concept Phase. Descriptive information that fits the FD should be entered into the FD section of the Progress Notebook; other information should be entered into other sections of the Progress Notebook as defined by the PM.

Problem definition information to be entered includes:

- A definition of the problem and its associated categories and reasoning subcomponents, including, if appropriate, why the problem space is shallower than described during the Initiation Phase.
- A description of each problem category.
- A description of the category ranking process and how the first category was selected.
- A description of the continuum of categories and the expected order of their implementation for the demonstration prototyping effort, identifying any categories or reasoning subcomponents that cannot be implemented.
- A description of the examples cases explored in detail.
- A description of how each high-risk ES technology need will be handled and a discussion of remaining risks.
- A description of the results of exploring alternative programming and representation methods and recommendations.

- A description of user interface needs and design recommendations.
- Descriptions of the knowledge sources and associated risks.

5.4.1.2 Exploring Integration with AIS and Special Devices (T2)

Integration of the ES with AISs, embedding an ES within an AIS, and interfacing ESs to special devices pose serious potential risk factors. (Special devices include sensors; special window handlers; voice input and/or output; touch panels; special graphics boxes; and ES access to device controllers in the embedding system, e.g., robotic controls.)

The risk is manageable if a successful production example can be found and emulated or if integration feasibility can be demonstrated. The issues and risks identified in the Concept Phase now need to be more deeply explored, although the actual interfacing activity is deferred to the Development and Deployment Phases.

Schoen and Sykes [Schoen 1987] say:

> Frequently the primary focus during the early stages of the development of an AI system is to determine that the technology risk is adequately bound and that the final system will justify its cost. Under these circumstances, it is best to defer most of the integration activity, beyond demonstration of feasibility, until later when the technical risk of integration is generally less and the potential value of the system can be measured without connecting it to on-line data sources. Deferring a capability for on-line operation till later does not mean that providing on-line operation can be done immediately or is inexpensive. In fact, it is likely that integration of an AI system may involve considerably more effort and cost than has been expended in the development of the original prototype. It may also be that a selected supplier of AI technology has given less attention to integration issues than might be expected and the final integration effort may provide some surprises.

More than one integration solution may need to be considered if the AIS will be undergoing redesign and development during the same period in which the ES is under development. It may also turn out to be less costly to change an AIS component (e.g., provide a standard network) than to build special-purpose software extensions to the ES.

The KE should work with one or more ADP specialists to develop integration designs. The first step in this effort should be to look for an existing production solution that could be used or one that is similar to the problem at hand. Failures should also be explored in order to gain from lessons learned and to better assess the risk. If appropriate for risk reduction, specific ES tools that have been successfully integrated with AISs should be identified as well as any underlying language requirements (e.g., use of C or Pascal).

A conceptual prototype should be developed for each AIS and special device requiring ES interfacing/embedding. Since the ES tool used for deployment is undecided at this time, the prototype may be based on an actual or example tool (e.g., the in-house tool being used during this phase). The prototype may be: a description of an existing production solution that will be emulated; a design and walk through; a limited demonstration of the interface using simulated data; or a real demonstration of the interface. Each interface will be re-evaluated as to risk, and risk management techniques will be recommended.

A checklist of integration facts should be made for each AIS or device to which the ES will interface, and this should be documented in the Progress Notebook. The check list consists of:

- A detailed list of the different AIS system components:
  - Network type or special purpose interface description
  - Data protocol/data exchange definition
  - Databases, files, applications furnishing data
  - DBMS, file system
  - Operating system description
  - Hardware configuration: computer, storage system, special I/O, graphics interfaces

- A description of the concept prototype, risks and risk management plan. For embedded ESs, specific internal interfaces should be described:
  - How the AIS will call the ES component
  - How the ES component will call AIS components (user interface, data interface, etc.)
- An integration plan describing the configuration of the AIS or special device, and discussing labor, elapsed time, cost, estimated risk, and rationale. If any component is scheduled to be changed, discuss change, schedule, impact, and risk.

5.4.1.3 Exploring Security and Integrity Needs (T3)

Security and integrity needs are issues of technical and managerial concern. After the technical needs have been defined and the security classification of the ES has been determined, the PM should consult DoD security policy, documentation, and personnel to determine the security level required for ES development and for deployment of the ES product. These security levels define system requirements that will be an integral part of the ES design and architecture. Again, although no ES tool has been selected at this time, ES tools that support the security and integrity needs should be identified.

In some software development efforts of secure systems, exploratory prototypes are developed using sanitized data. This may not be a wise alternative for ES development even if the problem space can be bounded to define an unclassified subproblem. It may be unwise because knowledge acquisition is a difficult process and having to consider the classification of each piece of knowledge would make it even more so. On the other hand, the PM may determine that a testbed using unclassified, sanitized data is a suitable surrogate and choose this lower-cost option. The determination of whether a sanitized testbed is useful is application-dependent.

Security will affect the deployment environment; a secure system architecture is needed that includes all interfacing/embedding AIS(s), the ES workstation(s), and the interfaces between system components.

Building a single security level or system high ES may be doable, but providing multilevel database security is still a research issue. Even its solution may continue to pose problems for ESs. Morgenstern notes that for multilevel rule-based KBSs, classification will need to be applied to the rules as well as to the data [Morgenstern 1987]. If rules are classified at a higher level than data, it is possible that the results produced by the system may depend upon the user's authorization level. Thus users at different security levels would get different responses to the same problem.

Integrity becomes a critical issue when the system needs to safeguard against errors in the knowledge base that might result in true disasters if the wrong decisions were made, e.g., when controlling a chemical processing plant or a dangerous machine. Methods of ensuring a high level of integrity are expensive; the most commonly used are formal methods for (1) verifying small critical sections of an application or (2) building several independent implementations of critical sections and using a voting procedure at runtime to determine output.

Concept Prototypes demonstrating the satisfaction of security and integrity needs should be developed. A prototype may be: a description of an existing production solution that will be emulated; a design and walk through; or a limited but actual demonstration. The security risk should be re-evaluated and risk management techniques recommended.

5.4.2 Defining Other Application Characteristics (T4)

Two other application characteristics that need to be considered by the technical team are maintenance needs and ES performance needs.

Maintenance Needs. Maintenance needs are of technical and managerial concern. The technical considerations include: estimate of maintenance needs for the next phase and simplification of maintenance levels and the KBA role after deployment.

Maintenance needs for the Definition/Design Phase can be estimated based on the estimated size of the project, number of technical staff, and number and size of ES tools, support tools, and workstations that will be needed. This information should be entered in the Progress Notebook.

Section 5.3.3.7 discussed the four levels of post-deployment maintenance that may be required: Level 1—integration; Level 2—workstation HW/SW; Level 3—ES tool; and Level 4—ES. Not much can be done to simplify maintenance at Levels 1 and 4, but Levels 2 and 3 can be improved if it is technically feasible and cost effective to:

- Select a production ES tool that is written in a conventional programming language such as C (several of the current tools either are or will be implemented in C), or implement the production system in a conventional programming language (e.g., C or Ada).
- Deploy the ES on a common workstation.

The feasibility, cost, and recommendations should be documented in the Progress Notebook and be used by the PM as input to the maintenance plans.

The size of the KBA effort is a function of several factors including:

- The expected size and frequency of knowledge base changes once the system has stabilized.
- The size and frequency of changes in the underlying data sources.
- The size and dispersal of the user community.
- The structure of the user community; is there a single user group that will be using copies of the ES, or are there multiple user groups who will need specialized versions of the ES to meet their needs?
- If there are multiple groups of users with specialized versions of the ES, are all versions to be controlled by a single KBA or will each group have its own KBA?

Analyze the user community needs, and estimate the effort required of one or more KBAs. If there are to be multiple KBAs, it may be difficult to coordinate new releases of the tool or workstation software. The maintenance community may prefer to deal with each user community on a separate basis (as if each had a different ES application). Perhaps in the Post-Deployment Phase each user community will actually have separate ESs that will be supported accordingly.

Document in the Progress Notebook the analysis of the user community and maintenance community interrelations after deployment and the estimated effort required of one or more KBAs, and furnish this information to the PM for the post-deployment maintenance plans.

ES Performance Needs. Specification of articulate and reasonable performance and timing needs will be an aid in:

- Writing the tool specifications and the SOW.
- Evaluating the next three prototypes in order to make performance-related design modifications.
- Ensuring the success of the system in the end-user environment (if the requirements are met).

Performance characteristics include:

- Size and complexity of the knowledge base to be handled and size of external and internal databases.
- Required capacity of the ES tool in terms of the knowledge base and the internal and external databases.
- Required bandwidth for communications networks.

- Timeliness (in terms of range of seconds) of:
  - Performance of certain tasks.
  - Response to user.
  - Data retrieval.
  - Specific needs, e.g., displaying a chart, drawing graphics.
  - Embedded system: ES response to information system and information system response to ES.
- Availability of the ES stated in terms of percentage of time available during a stated period of use.
- Reliability: frequency of occurrence of errors in knowledge base and operation during a stated period of use.

The KE should estimate the performance characteristics in cooperation with the expert, ADP specialists, and end-users while building and experimenting with the conceptual prototypes. These characteristics may undergo change as later prototypes are developed and the problem and solution better defined. Document the estimated performance characteristics in the FD section of the Progress Notebook.

5.4.3 Developing the Application Checklist (T5)

Develop an application checklist derived from conceptual prototyping, maintenance characteristics, performance characteristics, configuration management needs, test and evaluation needs, and deployment needs.

Try to organize the needs into meaningful categories that can be prioritized and weighted as an aid in specifying tools, determining acquisition strategy, and writing the SOW.

An example checklist of desired application characteristics follows.

I. Application performance needs
   - A. Size of system for Demonstration Prototype and size of deployed system
     1. Estimated knowledge-base size and complexity (e.g., rules, bytes)
     2. Estimated internal and external database size (bytes)
   - B. Acceptable response-time ranges for prototypes and deployed system (for prototypes state maximum response time allowable for adequate user testing and evaluation)
     1. Response to specific reasoning subcomponents or cases
     2. Response to user
     3. Response to external data retrieval
     4. Response to specific needs (e.g., drawing a chart, displaying graphics)
     5. Responsiveness of AIS to ES calls, and responsiveness of ES to AIS calls
   - C. Availability of ES
   - D. Reliability, and capability to fail and recover

II. ES performance-enhancing techniques
   - A. Optimization
     1. Intelligent look-ahead (strains ES capability)
     2. Wisely moving knowledge and data in and out of memory (strains ES capability)
     3. Rule compilation (as opposed to interpretation)
   - B. Real-time control
     1. Access to garbage collection (may strain ES capability)
     2. Need for control over time constraints (may strain ES capability)
   - C. Ability to access and change the internal capabilities of the tool
     1. Tool parameter-setting functions
     2. Access to source code

III. KBS representational power
- A. Arithmetic processing
- B. Logical processing
- C. Handling uncertainties (may strain ES capability)
- D. Inference control
  1. Iteration
  2. Forward/backward chaining
  3. Conflict resolution
  4. Inheritance
  5. Truth maintenance (may strain ES capability)
  6. Hypothetical reasoning
- E. Meta-knowledge: knowledge about knowledge, e.g., rules controlling inference (strains ES capability)
- F. Representation: rules, frames, procedures, objects
- G. Rule induction (may strain ES capability)

IV. End-user interface support needs (including special I/O devices)
- A. Ordinary presentation needs: text, graphics, windows, forms, mouse
- B. Extraordinary (special I/O device) needs
  1. Sensor devices
  2. Special window handling
  3. Voice I/O
  4. Touch panels
  5. Special graphics
  6. KBS access to device controllers in embedding system (e.g., robotic controls)
- C. Explanation
  1. Execution trace
  2. Hooks for programmer-written explanation
  3. Knowledge base browsing
- D. Training aids: capability for integrating computer-aided instruction, e.g., integrated tutorials (strains ES capability)
- E. On-line documentation
- F. Support for multiple end-users of same knowledge base (strains ES capability)

V. Developer-user-interface support needs (including special I/O devices)
- A. Ordinary presentation needs: text, graphics, windows, forms, mouse
- B. Extraordinary (special I/O device) needs
  1. Sensor devices
  2. Special window handling
  3. Voice I/O
  4. Touch panels
  5. Special graphics
  6. KBS access to device controllers in embedding system (e.g., robotic controls)
- C. Explanation
  1. Execution trace
  2. Hooks for programmer-written explanation
  3. Knowledge base browsing
- D. Training aids: capability for integrating computer-aided instruction, e.g., integrated tutorials (strains ES capability)
- E. On-line documentation
- F. Support for multiple end-users of same knowledge base (strains ES capability)
- G. Support for incremental development

H. Debugging tools, tracing
I. Metering capability (e.g., active values)

VI. Knowledge acquisition support needs
A. Rule induction (may strain ES capability)
B. Model building aids
C. Knowledge-base syntax checking
D. Consistency checking across rules (strains ES capability)
E. Knowledge classification checking (strains ES capability)
F. Knowledge-base editing
1. Structure editors
2. Graphic rule displays

VII. Integration needs
A. Integration with interfacing/embedding AIS
1. Description of interfacing AISs
2. Description of embedding AISs
B. Integration with other off-the-shelf workstation packages
1. Editor/word processor
2. Spreadsheet
3. DBMS
4. Optimization tools (e.g., math, statistical programming packages)
C. General integration capabilities
1. Calling other languages
2. Interprocess calls

VIII. Configuration management needs (strains ES capabilities)
A. Management of module checkout, returns, testing
B. Compilation dependencies support
C. Version control support
D. Support for multiple developers on separate workstations

IX. Test and evaluation needs (may strain ES capabilities)
A. Tool support for maintaining database of test cases
B. Tool support for timing performance tests
C. Batch running of test cases (identifying run and saving selected states, variables, results)

X. Security and integrity needs
A. Level of security required
B. Physical security
C. Workstation operating-system requirements
D. User identification and authorization (may strain ES capabilities)
E. Marking internal data and knowledge (may strain ES capabilities)
F. Marking output
G. Integrity: formal verification techniques (strains ES capabilities)

XI. Maintenance needs
A. Support for diagnostic testing (strains ES capabilities)
B. On-line manual support
C. Software configuration management support for compilation dependency checking, version control (strains ES capabilities)

XII. Automated documentation needs
A. Code/data annotation (may strain ES capabilities)
B. Assumption/rationale history (may strain ES capabilities)

XIII. Life-cycle development needs
A. Development environment
1. Preferred tool(s), workstation(s), implementation language(s)
2. Need separate development tool (e.g., deployed system will be developed in conventional language)

B. Deployment environment
   1. Workstation type(s), operating system/other software, preferred implementation language, number of workstations
   2. Requirement for direct modification of deployed system
   3. Requirement for protection of deployed-system knowledge base
   4. Capability to transfer/rehost to another type of deployed system

Weight the application characteristics presented in the checklist (e.g., on a scale of 1–5, where 5 is the most important) to reflect needs of the *Demonstration Prototype* (design and development tool), *Testbed Prototype* (deployment tool, possibly with design support), and *Operational Prototype* and deployed system (deployed version of testbed tool). This will result in a separate profile for each prototype stage. For each profile, identify the critical characteristics (e.g., those rated 4 or 5) that *must* be met by the ES and that will be drivers in the ES tool evaluation and selection. Select these carefully. Difficulty or inability to meet these characteristics implies high risk and may result in termination of the project effort.

Document the Progress Notebook to reflect the checklist profiles and critical characteristics for each profile, along with risk analysis and recommendations for critical characteristics that are high-risk items.

5.4.4 Holding the Review to Determine FD Adequacy (J1)

Determining the adequacy of the functional description is a joint technical and management activity. The review should include the technical staff, the PM, representatives from the user community (expert(s), end-users, and management), an ADP systems person, a security person (if needed), and a representative from the maintenance community. This is an in-depth review held to identify all major problems in order to put additional effort into solving them before the Milestone 1 review.

A detailed walk through the functional description should be made using prototype demonstrations as an aid in understanding. The objective is to determine that all needs and issues making up the core of the problem have been defined in enough detail to proceed to the next phase. Particular emphasis should be placed on areas identified as risk items. Risk resolution measures should be presented for each item in detail and critiqued.

The review will result in a green light for the project if: (1) the user community is satisfied and enthusiastic about the accuracy of the problem description, including the user interface and how well the project fits their needs; (2) the ADP and security people agree that the proposed architecture will adequately handle both integration and security needs; and (3) the maintenance representative is satisfied that the maintenance organization can furnish or realistically plan to furnish the necessary support.

5.4.5 Developing an Acquisition Strategy (J2)

Development of an acquisition strategy that effectively integrates the technical, business, and management elements of the project is a joint technical and management effort. The technical contribution will provide (1) estimates of the ES tool needs and costs and other SW/HW needs and costs based on the FD and conceptual prototypes and (2) estimates of development staff size and composition over the project life cycle including staff training. The PM will use the technical contributions as input in the evaluation of alternative acquisition strategies.

To estimate ES tool needs and costs, the KE should use the Demonstration, Testbed, and Operational Prototype checklist profiles to evaluate and select representative ES tools. This will involve various trade-offs such as:

- *Different ES tools vs the same ES tools.* The cost of using different ES tools for design and deployment may be high but productivity increase and reduced risk may save overall. In contrast, there may be a great advantage in selecting an ES tool family to span design to deployment, even though one or more of the tool members used in different stages may be less adequate than a tool outside the family.

- *Easily maintained ES tool vs easily maintained knowledge base.* Selecting an easily maintained ES tool and workstation with high performance for deployment (i.e., selecting a tool written in a conventional programming language and a standard workstation) may appear to save operation and maintenance (O&M) costs during the life cycle. However if the KBA has to maintain a frequently changing knowledge base without adequate tool support, this may end up increasing O&M costs beyond those saved.
- *Low-cost ES tool vs ES tool with growth potential.* Selecting an ES tool with a low licensing fee because the user community is large may result in a limitation on the growth of the knowledge base if (1) the tool/workstation is inadequate to handle a large knowledge base or (2) it can only handle it with increasing performance degradation.

Information about how to do tool evaluation and selection can be found in Appendix C of the RAND Note that contains additional appendices to this Guide [Kameny 1989]. The process and results should be documented in the Progress Notebook.

The PM should take the project size, estimated costs, and schedule into consideration when exploring different acquisition strategies. These include decisions about where the work will be performed and how to acquire the necessary tools.

The PM can choose from many ways to carry out the project and will have to study ideas such as:

- Performing the ES project in-house vs contracting it out;
- Performing the ES part of the project in-house and hiring specialists for integration, security, special device interfaces, etc.;
- If the project is very large, deciding the contractor for the Demonstration and/or Testbed Prototype efforts by holding a design runoff between the contractors;
- Letting a separate contract for the Definition/Design Phase, precluding the design contractor from bidding on later phases;
- Performing some of the phases in-house and contracting others out (e.g., design and definition in-house but development and deployment outside, or the reverse).

The PM also has many choices as to how to acquire the tools: letting the contractor order; ordering from a DoD preferred list through a DoD acquisition agent; or using government furnished equipment (GFE) in-house or furnished to the contractor(s). Likewise, staff training and maintenance during the development phases can be left up to the contractor or furnished by the government. The PM should discuss with his contracting officer how best to contract for a project that is incrementally acquired and funded.

Finally, the PM should decide on an acquisition strategy that will be presented at the Milestone 1 review. It should include:

- How the project effort will be carried out with respect to contractual vs in-house resources and, if contracted, recommended type of contract (e.g., cost plus fixed fee).
- Generically, what the tool specifications recommend as ES tools for the Definition/Design, Development, and Deployment Phases (not naming specific commercial ES tools).
- Detailed description of resource needs and acquisition plans for the Demonstration Prototype, including: implementation of the Demonstration Prototype; tool acquisition; training; user community participation; ADP, security, and maintenance community participation; maintenance support during prototype development; estimated cost and schedule.
- Rough estimate of cost and schedule for the other phases, with indications as to when additional acquisitions should begin.

In the Progress Notebook, document the analysis of the alternative strategies considered for acquisition and the selected strategy.

5.4.6 Writing the ES Tool Specification (T6)

After the acquisition strategy has been determined, the KE should write an ES tool specification that supports the acquisition strategy.

The application checklist profiles should be modified to reflect the new prioritization of application characteristics as a result of the acquisition strategy analysis. The specification will be based on the updated checklist profiles and should stress any special characteristics critical to carrying out the acquisition strategy analysis—e.g., that the ES tool should be written in a conventional programming language or that the design tool should belong to a family of tools, one member of which satisfies certain deployment characteristics.

The ES tool specification will be part of the SOW.

5.4.7 Writing the SOW for the Demonstration Prototype (T7)

After the Milestone 1 Management Review and the project is given the go-ahead, an SOW should be written whether the project will be supported in-house or implemented under contract.

The SOW should include:

- The functional description of the problem, taken from the Progress Notebook, including:
  - Scope of the problem to be addressed in the Demonstration Prototype and, if applicable, a broader scope with indications as to when the additional breadth will be provided (e.g., post-deployment follow-ons). Include descriptions and results for the conceptual prototypes.
  - ES tool specifications including the application checklist and profiles.
  - Environment description: current and future AIS integration, embedding, special device interfaces, security requirements, and data requirements. Include descriptions and results from the conceptual prototypes.
  - Information about the user community (availability of experts, end-users, and test data), about the maintenance community, and about the availability of government ADP and security personnel to aid in definition/design.
  - Discussion of all risk factors and open issues, including the results of risk analysis and recommendations for resolving the risks.
- Qualification requirements of the contractor, including:
  - Expert-system development experience by project.
    - Description of project, including current state of effort.
    - Project size in terms of ES tool used.
    - Project size in terms of staff, person-years, and cost.
    - Client contact person.
  - System integration experience by project.
    - Description of project, including current state of effort.
    - Project size in terms of person-years and cost.
    - Client contact person.
  - Relevant domain experience.
    - Projects performed.
    - Experience of identified personnel.
  - Experience levels with specific ES tools and AI machines, and an indication of those in-house and available for use.
  - Size and experience of the ES staff.
  - Identification of key personnel available to work on the project.
- Incremental development of the Demonstration Prototype.
  - Well-defined plan and in-process review (IPR) goal for first iteration.
  - Current FD definition of the desired finalized Demonstration Prototype.

  - Business plans including ability to incrementally fund and modify schedule and plans to accommodate refinements as a result of IPRs.
- Intended scope of SOW. SOW will cover the Demonstration Prototype effort only or a phased program of which the Demonstration Prototype effort will be the first phase. If phased program, indicate how acquisition of additional phases will be handled.

6. DEFINITION/DESIGN PHASE, DEMONSTRATION PROTOTYPE[4]

6.1 INTRODUCTION

Figure 6.1 shows the place in the expert system development process of the Definition/Design Phase, Demonstration Prototype. Its major technical and management activities are shown in Figure 6.2. The gray boxes there indicate the process for handling an architecture change. The Definition/Design Phase consists of two prototype efforts, the Demonstration Prototype described in this chapter and the Testbed Prototype described in Chapter 7. This phase is concerned with producing a complete (but stand-alone) definition of the problem and design/implementation of the problem solution. The Demonstration Prototype uses an ES tool selected primarily for its flexible support of exploration and design. The Testbed Prototype is basically a redesign and re-implementation of the Demonstration Prototype using a version of the ES tool that will be part of the deployed ES. (The acquisition strategy developed in the Concept Phase took into account the desirability for both prototypes to use the same ES tool or ES tools belonging to the same family.)

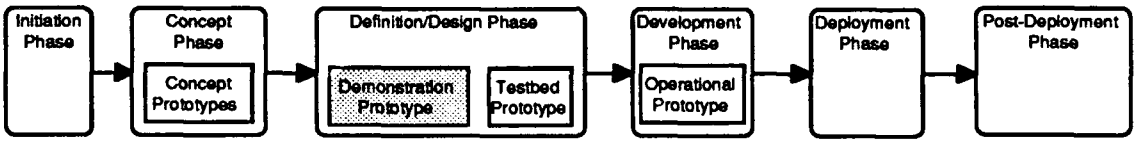


Figure 6.1—Expert system development process, Demonstration Prototype

The Demonstration Prototype is the second of the four prototype stages in the development of an ES using a risk management approach. It focuses on rapid design and prototyping of the problem solution in a stand-alone mode using an ES tool selected primarily for its capabilities to support the Knowledge Engineer (KE) in his/her design effort. The ES tool selected should offer extensive support for debugging, graphics, knowledge acquisition, user interface development, and multiple modes of reasoning and knowledge representation.

This phase begins after the acquisition strategy (decided upon and implemented in the Concept Phase) has resulted in an in-house commitment or contractual choice to carry out the SOW.

Initial management activities (M1) include: developing a management plan; acquiring the project staff, ES tools and environment, and necessary training for expert(s), end-users, and project staff; carrying out plans developed for QA, configuration management, security, and maintenance; and defining the documentation requirements for the Demonstration Prototype effort. After this, the PM and technical staff will jointly define a detailed T&E plan for the Demonstration Prototype (J1). Ongoing activities (M2) include: carrying out the project plan and modifications made to it as a result of the prototyping effort; monitoring QA, configuration management, T&E, training, risk management, security, and project support; and facilitating exchanges across the developer, user, maintenance, ADP, and security communities. The project manager (PM) will conduct in-process reviews (IPRs) (M5) at the conclusion of each prototype iteration or by demand as major unanticipated problems arise. Finally, the FD will be presented at a Milestone 2 review (M4) along with all plans and

[4]See Appendix A for definitions of acronyms and terms used in this chapter.

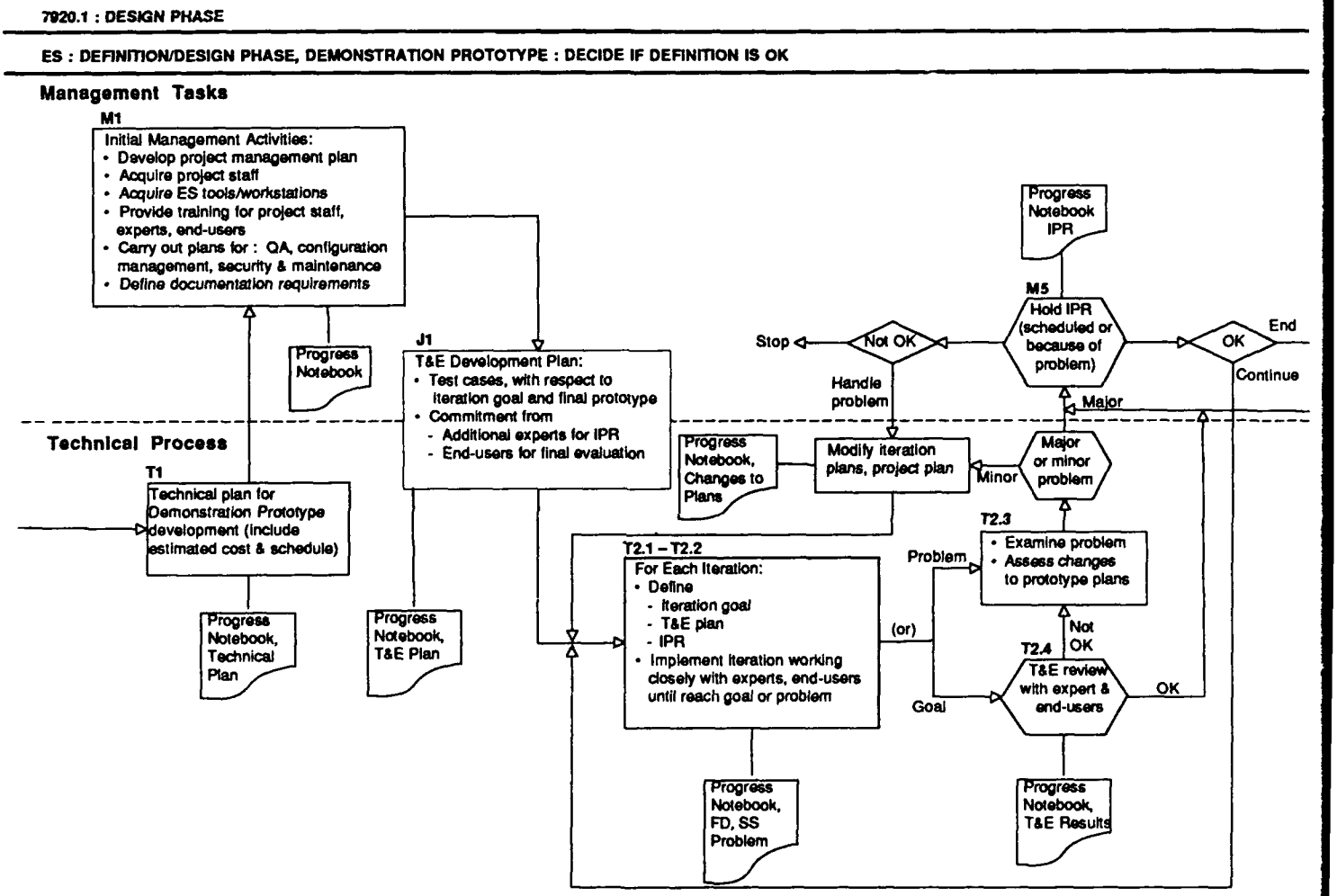
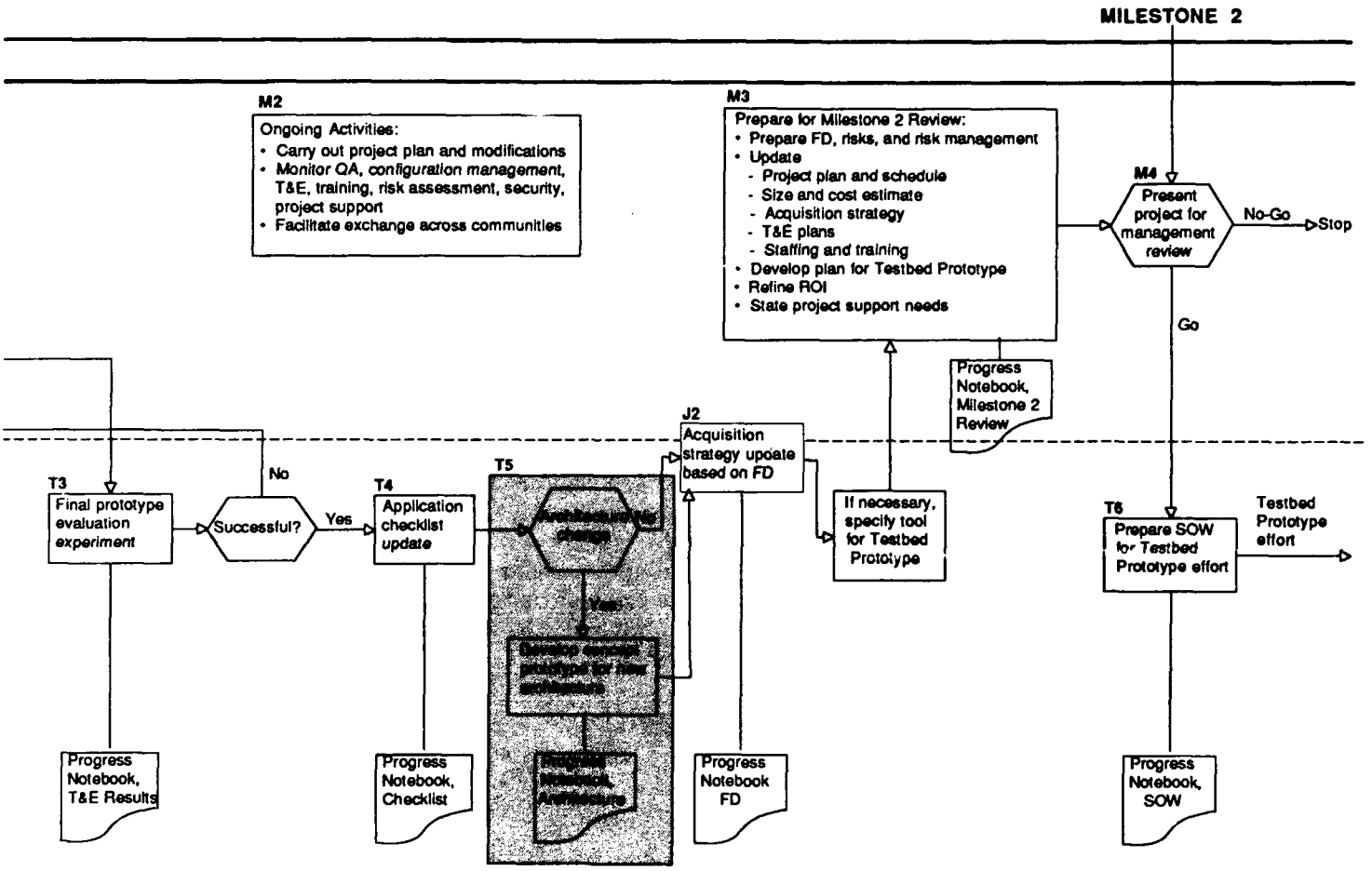


Figure 6.2—Definition/Design Phase, Demonstration Prototype


MILESTONE 2
M2
Ongoing Activities:
• Carry out project plan and modifications
• Monitor QA, configuration management, T&E, training, risk assessment, security, project support
• Facilitate exchange across communities
M3
Prepare for Milestone 2 Review:
• Prepare FD, risks, and risk management
• Update
- Project plan and schedule
- Size and cost estimate
- Acquisition strategy
- T&E plans
- Staffing and training
• Develop plan for Testbed Prototype
• Refine ROI
• State project support needs
M4
Present project for management review
No-Go
Stop
Go
Progress Notebook, Milestone 2 Review
J2
Acquisition strategy update based on FD
T3
Final prototype evaluation experiment
Successful?
No
Yes
T4
Application checklist update
T5
If necessary, specify tool for Testbed Prototype
T6
Prepare SOW for Testbed Prototype effort
Testbed Prototype effort
Progress Notebook, T&E Results
Progress Notebook, Checklist
Progress Notebook FD
Progress Notebook, SOW

reconfigured plans necessary to continue to the Testbed Prototype effort. The result of the Milestone 2 review is either: to continue the project by moving on to development of the Testbed Prototype; to iterate back into the Demonstration Prototype activity because the problem definition and design are inadequate; or to terminate the project.

The technical process begins by defining the technical plan (T1) for developing the Demonstration Prototype product. The product should be defined in terms of its functionality, with cost and schedule constraints clearly stated. The next step is a joint technical/management preparation of the T&E plan (J1). This plan will specifically define the test database, obtain commitments from additional experts to review the prototype at IPRs, and obtain commitments of selected end-users for the final evaluation of the Demonstration Prototype.

The iterative development of the Demonstration Prototype (T2) is next. Each iteration must have a well-defined objective, T&E plan, and IPR. A major problem arising during an iteration should trigger an unscheduled IPR. Each IPR (M5) will determine if the project is proceeding well in meeting its plan, if the plans should be changed, if the prototype is completed, or if the project should be terminated. The KE and expert should work closely together to build the prototype. T&E results will be reviewed by the expert and end-users. The IPR will be attended by the expert and end-users and the results reviewed and critiqued by additional expert(s).

At the end of the prototype development, the final T&E will include the experimental use and evaluation of the Demonstration Prototype by a small group of end-users who have not been involved in the prototype effort. The application checklist will be updated (T4) to reflect changes to the problem definition, AIS architecture, I/O devices, and security and telecommunication needs. Architecture changes (T5) may necessitate building new Concept Prototypes to show how integration, interfacing, security, and communication changes will be handled in a modified architecture. Changes in the application requirements may necessitate re-evaluation of the acquisition strategy (J2), resulting in re-evaluation and selection of the deployed ES tool and a new ES tool specification for the Testbed Prototype. The FD, ES tool specification, and recommendations are fed into management preparation for a Milestone 2 review (M3). The last technical task is to write an SOW for the Testbed Prototype (T6).

This chapter is divided into four sections: Section 6.1, this section, is an introduction; Section 6.2 presents a risk containment table; Section 6.3 describes the management activities; and Section 6.4 describes the technical process.

6.2 RISK CONTAINMENT

Table 6.1 is an example prioritized list of major risk items that need to be identified and addressed during the Demonstration Prototype effort. Since applications vary, the project manager should develop his/her own list of the top ten risk items.

6.3 MANAGEMENT ACTIVITIES

The management activities are organized into:

Table 6.1

AN EXAMPLE PRIORITIZED LIST OF DEMONSTRATION PROTOTYPE RISK ITEMS

	Risk Item	Risk Management Techniques
1.	Personnel shortfalls	Staff this phase with top talent; pre-schedule key people; enact team building and training
2.	Lack of user community participation	Obtain commitment for qualified expert(s) to work with KE; additional experts for review of ES; sufficient number of end-users; test cases with criteria
3.	Straining of ES capabilities	Scope application to eliminate hard problems; demonstrate low-risk doable solutions in Demonstration Prototype; avoid extending the problem scope by putting off enhancements until post deployment
4.	Effect of changes in requirements on validity of problem to user mission	Evaluate mission needs whenever considering major change to problem scope
5.	Changes to relevant AIS(s), I/O devices, security, and telecommunications	Detect external system changes early on; evaluate changes; refine concept prototype if necessary to demonstrate new architecture
6.	Inefficiency and costliness of monthly reviews and unscheduled IPRs	Hold informal monthly reviews with little paper work; ensure quick unscheduled IPRs with minimum formal preparation, restrict attendees to PM's discretion
7.	Modification of existing acquisition strategy	Recognize requirement changes and project experiences that may affect acquisition strategy; perform cost/benefit analysis of changing the strategy
8.	Unrealistic schedule and expectations	Be realistic in plan and schedule; prepare user community for incremental acquisition/rapid prototyping approach requiring some flexibility in changing plan, schedule, and funding
9.	Documentation: maintaining the Progress Notebook; problems with such maintenance	Assign project librarian to be responsible for maintaining all documentation; define how different types of information not in FD and SS will be documented in Progress Notebook; oversee and review documentation; ensure documentation is validated
10.	Difficulty in doing cost benefit/ROI	Compare actual test-case answers with ES responses for better estimate of benefit

6.3.1 Initial Management Activities (M1)

The initial management activities are: (1) to develop a management plan; (2) to acquire the project staff, ES tools and environment, and necessary training for expert(s), end-users, and project staff; (3) to carry out plans developed for QA, configuration management, security, and maintenance; and (4) to define the documentation requirements for the Demonstration Prototype effort.

Management Plan. A management plan should be prepared that includes: technical plan milestones; schedule for acquiring staff; schedule for acquiring ES tools and equipment; training schedule (for staff, experts, and end-users); documentation schedule; maintenance schedule; IPR schedule; and the schedule of monthly meetings to review progress and risk management.

Staffing, Training, and Equipment. The technical staff requirements were addressed in the SOW and should be reviewed with the KE or technical manager and modified if necessary. Staff acquisition and training are of prime concern, and the technical schedule should reflect a realistic estimate of when trained staff will be available to work. If the work is being done in-house and staff is unavailable, the PM may want to consider hiring qualified consultants. Expert system staffing is discussed in Section 5.3.3.5 and training in Section 5.3.3.8. Training is also required for participants from the user community. The types of training necessary for experts and end-users are discussed in Section 5.3.3.8.

The timely acquisition of ES tools and environment (e.g., workstations) is also critical. If the technical staff is being trained locally, then it is very important for them to have the tools available during the training period.

Carrying Out Plans. It is the PM's responsibility to see that plans developed for QA, configuration management, security, and maintenance in the Concept Phase are carried out.

- The PM should ensure that technical staff are informed and/or trained as to QA guidelines and configuration management procedures.
- The project librarian may need special training and/or computer aids in order to best carry out the configuration management functions.
- If the Demonstration Prototype is a classified project, then the PM must carry out the security plan with regard to: physical environment (including workstations and disks); clearances for staff members, experts, users, ES/workstation maintenance personnel (if required), and others as required; labeling of inputs and outputs; user/developer access and authorization procedures; etc.
- The PM must plan for ES tool/workstation installation and checkout; ongoing maintenance of equipment; and either vendor support for the ES tool or a resident tool smith.

Documentation Requirements. The PM should define the documentation requirements for the demonstration prototyping effort and make them known to the technical staff and documented in the Progress Notebook (PN). The minimum amount of documentation is shown in Figure 6.2 and was derived from Figure 3.2, "Automated information systems life-cycle documentation" according to Draft DoD Standard 7935.1, 15 January 1988. The documentation requirements should identify specific sections in the PN for the Functional Description (FD), the System/Subsystem Specification (SS), and the Testbed Prototype SOW. The PM should further designate sections of the Progress Notebook for information, issues, and problems not provided for in the FD and SS. Further sections to be considered are: project management plan; project technical plan; T&E plan and testing results; "problem" descriptions and resolutions; application checklist, prioritized list, and profiles; and reports on IPRs and monthly management meetings.

The PM should designate a project librarian who will be responsible for maintaining the documentation.

6.3.2 Development of T&E Plan for Demonstration Prototype (J1)

Developing the T&E plan for the Demonstration Prototype is a joint technical and management activity. It is addressed in Section 6.4.2 as part of the technical process.

6.3.3 Ongoing Management Activities (M2)

Ongoing management activities include: carrying out the project plan and modifications made to it as a result of the prototyping effort; monitoring QA, configuration

management, T&E, training, risk management, security, and project needs; and facilitating exchanges across the developer, user, maintenance, ADP, and security communities.

Managing the Project Plan. The objective of the Demonstration Prototype is to explore the problem by defining, designing, and implementing a stand-alone ES built using an ES tool selected for its flexible support in design. Built into the demonstration prototyping concept is the need to easily and inexpensively accommodate changes to the technical and management plans, costs, and schedules. This requires careful management oversight to detect deviations from goals, schedules, and costs without imposing excess management demands on the technical team. The two review techniques for achieving this are: informal monthly project reviews and more formal, in-process reviews (IPRs).

The monthly reviews should concentrate on progress, risk management, and problems that occurred during the previous month. The review could be in the form of a short written report, a discussion between the PM and KE and perhaps a user organization representative, or a meeting of all or selected members of the project staff.

The IPR can be either scheduled (at the end of each prototype development iteration—approximately 6-month intervals) or unscheduled (called because of a major problem). The scheduled IPR addresses whether the ES has met its goals, the T&E results, problems to be resolved, risk management, revisions to plans, and the next step (e.g., continue the prototype, go on to the next iteration, conclude the project). Attendees may include representatives from all interested communities and should, as a minimum, include the project staff, the user organization manager, and experts and end-users participating in the project development.

The unscheduled IPR is held to address a major problem; attendance should be decided by the PM with recommendations from the KE or technical leader.

Activities that Need Monitoring. In addition to monthly reviews and IPRs, the PM should:

- Direct the KE or technical leader in monitoring QA.
- Monitor the configuration management activity and address any management problems that occur (e.g., technical staff evading the process).
- Monitor the T&E activity addressing any problems with plan, acquisition of test cases, participation of the user community, and follow-up on technical response to results.
- Monitor all training activities by talking with people who have been trained and their managers to determine that the training is being done well and is sufficient. If not, the PM should look into making other arrangements.
- Monitor risks. Though these will be addressed in monthly meetings, some risk management should be assumed directly by the PM. This includes risks external to the stand-alone ES prototype, such as changes to an AIS or device that the deployed system will be integrated with or changes to security or telecommunication requirements. If such a change affects the overall architecture, the PM may ask that a concept prototype be developed to demonstrate a new solution.
- Monitor security policy and methods to ensure that they are being properly enforced.
- Monitor project support needs: maintenance of tools and equipment, office and laboratory space and usage, secretarial and documentation support, etc.

Facilitate Exchange across Communities. The PM should facilitate information exchange across user, developer, ADP, security, and maintenance communities by keeping them informed and inviting them to IPRs. In particular the PM should establish close ties with the user community and frequently discuss user satisfaction with the user organization manager as well as experts and users participating in the project. The PM has a responsibility to keep the user organization manager informed of project needs for expert and end-user participation, including training and T&E. The project cannot succeed unless a good relationship is established between the user and developer organizations and unless the user organization accepts responsibility for its part of the project effort.

As the Demonstration Prototype develops, the PM should brief the project to higher management to attract champions, and generate enthusiasm and support. As mentioned in the Concept Phase, use of a videotape presentation to demonstrate the ES prototype is highly recommended.

6.3.4 Acquisition Strategy Update (J2)

Updating the acquisition strategy so that it effectively integrates the technical, business, and management elements of the project is a joint technical and management effort. It is discussed in Section 6.4.7 as part of the technical process.

6.3.5 Preparing for the Milestone 2 Management Review (M3)

The main purpose of the Milestone 2 review is to show a sufficient and adequate definition of the problem solution, a prototype design, demonstrable implementation, and a low-risk approach for redesign and re-implementation in the testbed stage.

Information to be presented at the review includes:

- The functional definition as implemented in the finalized Demonstration Prototype.
- Outstanding ES problems identified in the final end-user evaluation but whose solutions were not implemented.
- A discussion of AIS, security, and telecommunication risks and resolution.
- A discussion of other major risks and their resolution and management.
- An update on:
  - Rough size and cost.
  - Project acquisition strategy.
  - T&E plans.
  - Staffing and training.
- Project plan and schedule.
- Detailed plan and acquisition strategy for Testbed Prototype effort.
- Refined ROI.

Discussion of the FD. Prepare a high-level walk through the functional definition and show how it maps into the Demonstration Prototype design. Discuss any change in the problem scope since Milestone 1 and, if a change has occurred, the validity of the current FD to the user community's mission. Discuss risks of straining ES capabilities and preventing gold plating, and give examples of problems and solutions. A short videotape presentation of the finalized Demonstration Prototype may be an effective way to demonstrate the problem solution.

Discussion of Outstanding ES Problems. Describe outstanding problems resulting from the final end-user evaluation. The solution to each problem should be given, with a risk assessment and cost/benefit analysis to explain why the solution was not implemented.

Discussion of AIS, Security, Telecommunication and Other Risks and Resolution. Discuss all changes (actual, planned, and anticipated) in AIS and/or I/O devices to be integrated with the ES, security, and telecommunications that may necessitate changes to the ES architecture. Present the new Concept Prototype architecture that addresses these changes, and explain effects of the new architecture on the overall plan.

Update of Estimated Size and Cost. The finalized Demonstration Prototype is a definition, design, and implementation of the problem and can be used to generate a better size and cost estimate of the project than was given in the Milestone 1 review. The estimate may be more accurate if the same ES tool (or ES tools belonging to the same family) is used during the rest of the life-cycle development. The estimate should include an update on:

- The size of electronic and text source data and knowledge and an estimate of the effort needed to acquire them for the Testbed Prototype, the Operational Prototype, and the deployed system.

- Estimate of the system architecture costs (integration with AIS(s) and special devices, security, and communications) if there were changes.
- Estimated cost of plans for: maintenance during development, deployment, and post deployment (including the KBA maintaining the ES and changing knowledge base); training of development staff users, experts, and maintenance personnel; T&E; configuration management; and documentation.
- Estimated cost of project management activities and reviews.

Again, as in the Milestone 1 cost estimate, the costs of other similar efforts should be compared with the new cost estimate to ensure that it is not off by an order of magnitude or more.

Update of Project Acquisition Strategy. Acquisition strategy should include: a discussion (if appropriate) of changes to the ES tool acquisition strategy for the Testbed Prototype, amount of risk, and cost/benefit analysis; and a discussion of reasons for the selected acquisition strategy for the testbed stage. Any changes to plans for staffing, training, and deployment (including maintenance considerations) during the life-cycle phases should be presented.

Update of T&E Plans. Any changes to the T&E plans should be presented. Experience in developing the Demonstration Prototype may indicate a need for better criteria and for changes in the number and kind of test cases, and in the number of end-users and experts.

Update of Staffing and Training Needs. Experience in developing the Demonstration Prototype should result in a more accurate assessment of the type and number of staff required and the training requirements for additional staff or for all staff if a different ES tool is to be used in the Testbed Prototype effort.

Discussion of Project Plan and Schedule. An estimated plan and schedule should be developed for the project life cycle that is consistent with the new cost estimate. The plan and schedule should be firm through the Testbed Prototype effort and may be less precise for each successive phase.

Discussion of Detailed Plan and Acquisition Strategy for Testbed Prototype Effort. This plan will reflect the requirements that will be described in the Testbed Prototype SOW. It will include a breakdown of the Testbed Prototype into iterations. Each iteration will be described in terms of the functions it will provide and an estimate of effort level and schedule. Major Testbed Prototype risks will be identified, along with ways of resolving and managing them.

Refinement of the Cost Benefit/ROI. The PM should refine the cost benefit/ROI presented at the Milestone 1 Review based on changes made to the FD during the demonstration prototyping effort. Examples of changes in the problem definition that may affect cost benefit/ROI are: narrowing or expanding of the problem scope; and change in overall system design due to changes in relevant AIS(s), I/O devices, security, and telecommunications.

Section 6.4.2 discusses T&E and notes that, if the test cases being used in T&E are real cases that have been collected with their actual responses, then comparing the actual responses with the ES response may provide some definitive ROI information.

6.3.6 Holding the Milestone 2 Management Review (M4)

The last management activity of the Demonstration Prototype effort is to present the project at the Milestone 2 Management Review. Managers at all levels of the participating organizations should be encouraged to attend, as well as expert and end-user representatives from the user organization. As mentioned earlier, a videotape may be an effective way of describing the finalized Demonstration Prototype.

6.4 TECHNICAL PROCESS

The technical process of the Demonstration Prototype effort can be organized into the following tasks:

6.4.1 Technical Plan for Demonstration Prototype (T1)
6.4.2 Development of T&E Plan for Demonstration Prototype (J1)
6.4.3 Development of Demonstration Prototype (T2)
6.4.4 Finalized Prototype Evaluation Experiment (T3)
6.4.5 Application Checklist Update (T4)
6.4.6 Architecture Update (T5)
6.4.7 Acquisition Strategy Update (J2)
6.4.8 The SOW for Testbed Prototype (T6)

6.4.1 Technical Plan for Demonstration Prototype (T1)

The demonstration prototyping process is an in-depth exploration of the problem space (defined in Section 5.4.1.1) resulting in the definition, design, and implementation of the problem solution. The rapid prototyping process described below considers the fact that changes to the technical plan are expected during the process. The process easily accommodates change, but this should not be used as an excuse for ill-defined plans and goals. The better defined the product and development steps are, the more clear-cut will be the trade-offs and options to consider when problems arise and changes are required.

The technical plan consists of: a functional description of the finalized Demonstration Prototype and its overall cost and schedule; and a breakdown of the Demonstration Prototype by incremental effort. For each iteration, the prototype functional description, estimated cost, and schedule should be given.

The initial description of the finalized Demonstration Prototype is based on the Concept Phase Progress Notebook (the FD section) and the response to the SOW. Cost and schedule constraints should be clearly stated. If cost and schedule are the driving factors for this phase of the project, then the final Demonstration Prototype description should be an estimate of what can be accomplished within the cost and schedule constraints. The objective is to clearly and precisely define the end conditions of the prototyping process at this point in time.

The final Demonstration Prototype functional description will show the top-level ES architecture in terms of the general functionality within each major component and subcomponent (or module). Internal interfaces between components and modules will be identified. Since the Demonstration Prototype is a stand-alone ES, components or modules will be assigned the functions of emulating external interfaces to AIS(s) and/or special devices. If the operational ES is to be embedded within an AIS, then the user interface component should emulate the AIS user interface.

Section 5.4.1.1 gives an example of how parts of the problem solution can be prioritized in terms of categories to be handled. Some type of prioritization should be done to guide the decision about the parts of the problem to tackle first and those that should be put off until later iterations.

The scope of the problem to be addressed by the first iterative prototype should consist of enough of the high priority parts of the problem solution to produce an ES that (1) is demonstrable, (2) is meaningful to users who will be evaluating it, (3) does not encompass more of the problem solution than can be developed in approximately 6 months. For example, the first demonstration prototype iteration may cover in breadth several categories that were explored in the Concept Phase and in depth, those reasoning subcomponents that are necessary to support the selected categories. ("Categories" and "reasoning subcomponents" are defined in Section 5.4.1.1.) The categories and reasoning subcomponents

can then be mapped into the architecture to determine what components and modules will need to be developed. These will be developed only to the degree that is needed to support the limited problem description. Cost and schedule of this iteration will be determined after the functional description is determined.

Each iteration is a prototype that builds on the functionality of the previous iteration by adding new functionality or enhancements. The scope of each successive prototype iteration is determined by applying the same method used to determine the scope of the first iteration. The scope of the problem should be expanded to cover enough additional high priority parts of the problem to produce a demonstrable prototype that users will recognize as being a distinct advantage over the previous prototype and still be achievable within approximately 6 months. Cost and schedule of each iteration will be determined after the functional description is determined.

The technical plan will be documented in the Progress Notebook.

6.4.2 Development of T&E Plan for Demonstration Prototype (J1)

As soon as the technical plan is completed, the T&E team, with the help of the PM, should develop detailed T&E plans. This includes (1) acquiring the necessary database of test cases and determining the cases and criteria to be used to test the results of each prototype iteration, (2) commitment for additional qualified expert(s) to participate in IPRs, and (3) commitment for selected end-users to participate in the final evaluation experiment.

Test Cases. The number and structure of test cases is dependent on the application. However, the user community should be made aware of the importance of (1) providing test cases that cover the problem space and (2) furnishing enough of these to allow the development team and the T&E team to add fresh test cases for each prototype iteration. Unlike a conventional software system (e.g., a payroll package) where there is a "correct" answer for each test case, an ES response may be instead a response that is adequate and reasonable from the expert's point of view (reasonableness of the answer may be determined from the ES explanation). Test cases are the best way to test the coverage of the ES and how prone it is to failing at the edges of the problem space. In general, the more varied the test cases are, the better the ES will cover the problem space and demonstrate resilience at the edges—and it will be more likely to fail gracefully rather than to break down.

Test cases should be evaluated by experts, independently of the ES development, to decide what the desired response or set of possible responses to each case should be. If possible, each response should also include its rationale. These responses and their rationales are the criteria against which the ES will be judged. Agreement should be reached as to how the ES responses will be rated against the criteria. If the rationales are not included with the criteria, then the experts evaluating the system will have to judge whether the ES explanation for a response is acceptable. If the test cases are real cases that have been collected with their actual responses, then comparison of the actual responses with the ES responses could yield more definitive ROI information.

Additional Qualified Experts. Additional qualified experts should be made available to the project specifically to review and critique the T&E results presented at each iteration's IPR, and possibly to help in determining the test case criteria. In Chapter 4 we mentioned the importance of the expert's credibility. It is equally important that additional experts be recognized as credible and qualified authorities. The validity of the ES depends very much on the validity of the experts. The value of the ES will depend on how successfully it captures, uses, and explains expert knowledge acquired from the domain experts. The objective of T&E is to test the ES by evaluating its responses against the criteria. The validity of the criteria is beyond T&E—it is the responsibility of the user community, especially the experts.

Final Evaluation Experiment. The final T&E test of the prototype will be an evaluation experiment in which end-users, unfamiliar with the project, will receive end-user training and will be given a set of test cases to solve using the ES. Their performance should be recorded in detail (audiovisual equipment would be very useful here) and evaluated by the

T&E team, experts, and the project team. In order to carry out the experiment: evaluation criteria must be defined, end-user participation guaranteed by the user organization, end-user training scheduled, the experiment scheduled, and analysis planned for. In most cases, the results should feed into the Testbed Prototype as improvements to the user interface, but major problems may be found whose solutions need to be demonstrated through a continuation of the demonstration prototyping effort.

Detailed T&E Plan. A detailed T&E plan should be prepared showing schedule, participants, and amount of effort involved in: preparing test cases and criteria; running T&E prior to scheduled IPRs; and carrying out the final evaluation experiment. A version of the plan should be prepared for the manager of the user organization that will show the schedule of user participation required as part of the T&E process. If this amount of user participation cannot be supported and the amount that can be supported is insufficient, the T&E leader should inform the PM, and the PM may want to modify or terminate the project.

The T&E plan, user organization plan, test cases and criteria, problems and issues should be documented in the Progress Notebook.

6.4.3 Development of Demonstration Prototype (T2)

This section (T2) covers general blocks in Figure 6.2, as indicated in the following discussion.

General requirements for each Demonstration Prototype iteration are:

- A well-defined prototype plan, including a functional description of the prototype iteration to be achieved and cost and schedule constraints;
- An expert(s) available to work closely with the KE(s) to explore the problem space;
- Trained end-users available for T&E at the end of a prototype iteration;
- The test cases that will be used for development, each with a description of the functionality it is testing, the correct result, and the rationale supporting that result;
- A planned IPR using additional qualified experts to evaluate the T&E results.

Steps in the process of developing a prototype iteration are:

- 6.4.3.1 Plan for Prototype Iteration (T2.1)
- 6.4.3.2 Iteration Development (T2.2)
- 6.4.3.3 Handling Problems (T2.3)
- 6.4.3.4 Test and Evaluation of Iteration (T2.4)
- 6.4.3.5 In-Process Review of Iteration (M5)

6.4.3.1 Plan for Prototype Iteration (T2.1)

Each Demonstration Prototype iteration is an ES development effort with a well-defined goal, cost constraints and schedule constraints.

The prototype plan will include a functional description of the top-level ES architecture and its decomposition into components and modules, the test cases and criteria to be used in developing the ES, work assignments, and estimated schedule and cost to carry out the development.

The top-level architecture will describe the broad functions to be performed and the assignment of those functions to components and modules. Internal interfaces between components and modules will be identified; external interfaces will be emulated in components or modules. Each component and module should be identified as to whether it is to be: acquired intact from an earlier iteration; modified from an earlier iteration; or developed as a new ES element.

Work assignments and schedule will indicate who or which teams will be responsible for development of the various elements.

6.4.3.2 Iteration Development (T2.2)

The technical leader should be sure all technical personnel are aware of the QA requirements and should periodically review code to be sure the QA requirements are adhered to. For large projects, it is especially important to establish how the teams will coordinate their efforts when using the established configuration management practices.

Problem exploration and definition are accomplished in a way similar to that used in the development of Concept Prototypes discussed in Section 5.4.1.1, but applied at a deeper level using test cases and not necessarily pursued in the ordered fashion presented in Section 5.4.1.1. The steps described in that section are:

- Exploring the problem space
- Addressing risks in straining ES technology
- Exploring programming and representation methods
- Exploring the user interface
- Defining the knowledge sources
- Documenting the Progress Notebook

Defining the Problem. The problem space was defined in the Concept Phase by the KE working closely with the expert. Now the KE begins to explore the details of the functionality needed for new categories of cases in terms of the components and modules defined by the architecture. Again, he/she works with the expert to understand the heuristics and knowledge needed to support specific analytic steps. He/she will describe these so that the technical team can build them into the prototype and have the expert review/use/critique them to identify misunderstandings, errors, shortcomings, and insights. As additional categories of cases are defined and implemented, the KE and the expert will pay attention to some design implications. For example, some categories may require new or different reasoning subcomponents than those already defined. Should new modules be developed to handle these, or can they be fit into an existing module? Generality, internal complexity, and fewer modules may be traded off with specificity, internal simplicity, and more modules.

Potentially Extending the Problem Space. The KE should be aware of when exploration leads to extending the problem space rather than to refining it. This may occur if a category was loosely defined in the concept prototype and now is found to require significantly more subcomponents than the other categories, or to require source knowledge or data not previously identified, or to require the use of ES techniques that strain ES technology. This is a potential problem, and the KE should pursue it in enough detail to understand its implications and develop alternatives for handling it, one of which may be to remove the category from the application definition.

Straining ES Technology. The KE should identify as a potential problem any reasoning subcomponent that, as a result of more detailed exploration, needs a technical solution that would strain ES technology. The PM should assess the impact of not having the reasoning subcomponent.

Using Different Methods. The technical team may investigate the use of different programming and representation methods as well as architecture modifications as ways of improving the design. Again, any significant change should be identified as a "problem" and should be presented at an IPR as such.

Developing User Interface. User interface development may require interaction with the end-users as well as the expert. However, the KE should try to limit end-user involvement, reserving this for the T&E task. Limited user involvement may be adequate when the user interface is well-defined in the Concept Prototype and when the Demonstration Prototype iterations are short. However, if these conditions are not true, and the KE determines it to be a risk not to involve the end-users more, he should make the need known to the PM and the user organization and try to get more end-user participation.

Identifying Written and Electronic Data and Knowledge Needs. As the expert describes the heuristics and knowledge needed to support analytic steps, the KE should translate the needs into specific data/knowledge requirements and document these needs in the FD section of the Progress Notebook. As indicated above, any newly identified data/knowledge source or substantial change in the amount to be acquired and maintained should be recognized as a problem.

6.4.3.3 Handling Problems (T2.3)

A problem report should be prepared (and documented in the Progress Notebook) for any problem whose solution will substantially affect the plan. (This includes schedule and cost deviations.) The report will consist of: a definition of the problem, alternative solutions, and their respective effects on the current iteration plan and on the finalized Demonstration Prototype.

The KE and PM should decide when a problem is minor and can be handled by an internal change to the iteration plan, a minor change to the T&E plan, and possibly minor modifications to the finalized Demonstration Prototype. If so, the changes to the plans should be made and documented in the Progress Notebook and the iteration should continue.

If the KE and PM decide the problem is major, then an unscheduled IPR should be held at which time the problem will be presented for adjudication. (The IPR and its attendees are described in Section 6.4.3.5.)

Determining the End of the Iteration. Since deviation from cost and schedule is handled as a problem, the iteration will be ended when the iteration goal is met (the planned functional description is realized) or as a result of an unscheduled IPR.

6.4.3.4 Test and Evaluation of Iteration (T2.4)

Test and evaluation will be held at the end of the iteration. Both expert(s) and end-users will participate in T&E. There will be four sets of tests:

- Test cases from previous T&E runs will be rerun to establish the fact that the new ES supports the established T&E baseline. It is possible that a major change will induce some test cases to fail (e.g., if a category has been removed). Any failures should be individually evaluated by the KE and expert.
- New test cases will be run by the T&E team (with the help of a technical team member if necessary) and the responses recorded and analyzed with respect to the criteria.
- New test cases will be run by the expert and responses recorded and analyzed by the expert with respect to the criteria. If the rationale has not been provided, then the expert should judge how well the ES handled the reasoning behind the response.
- End-users will use the system to handle selected test cases with technical staff help if necessary. Their performances will be monitored and documented by the T&E staff. After using the system, the end-users will be interviewed about the problems they had and their suggestions for improvements. If possible, the end-user tests should be recorded using audiovisual equipment.

The T&E session should be reported in the Progress Notebook.

A problem report should be prepared for any recognized problem and documented in the Progress Notebook. The T&E leader, KE, and PM should decide if a problem is minor and can be handled by an internal change to the iteration plan, a minor change to the T&E plan (which includes rerunning the T&E after the changes have been made), and possibly minor modification to the finalized Demonstration Prototype. If so, the changes to the various plans should be made, documented in the Progress Notebook, and the iteration continued.

If the T&E leader, KE, and PM decide the problem is major, then the problem should be addressed at the scheduled IPR, which will be held immediately after the T&E report is prepared.

6.4.3.5 In-Process Review of Iteration (M5)

An IPR can be either scheduled or unscheduled. A scheduled IPR should be held immediately after the T&E to review the status of the project and, in particular, to evaluate the current ES prototype. An unscheduled IPR is called when a major problem is identified that may cause considerable change to the plans and schedule. Unscheduled IPRs may be called during a Demonstration Prototype iteration or as a result of the final evaluation test.

The people who should attend the IPR include: all project staff and managers, the user community project expert and additional expert(s), the end-users who are participating in the prototype development and/or the T&E, and user organization managers. Representatives from the ADP, security, and maintenance communities may also be invited.

Unscheduled IPR. The unscheduled IPR should address the current status of the project in meeting its goals and objectives, the problem that prompted the IPR, alternatives for its solution, and recommendations from the technical team. The PM (with recommendations from the KE) should decide whom to invite to the unscheduled IPR.

What to do about the problem will depend on a number of factors:

- How well the project is going.
- How important the problem solution is with respect to the worth of the application to the user community and its mission if the problem is not solved. If it is worth solving, then:
  - Availability of additional funding,
  - Acceptability of lengthening the schedule and/or finding additional staff.
- The amount of risk involved in the solution with respect to straining technology (ES, ADP, security, telecommunications, etc.).
- The degree of "gold plating" (capabilities outside the defined problem scope) required by the solution of the problem.

Alternative outcomes of the IPR are:

- To rescope the application to eliminate the need to solve the problem, to change relevant plans, and to continue the ES prototype.
- To obtain additional funding and a schedule extension and/or additional staff to solve the problem, to change relevant plans, and to continue the ES prototype.
- To declare the current ES iteration to be the finalized prototype. Stop the Demonstration Prototype effort at this point, and perform the end-of-iteration T&E, the final testing and other activities, and the Milestone 2 review.
- To declare the current ES iteration to be the finalized prototype and terminate the project.

The IPR proceedings and outcome should be reported in the Progress Notebook.

Scheduled IPR. The scheduled IPR should include:

- A review of the project's progress to date and how well it is meeting its objectives (this should include reference to all major and minor problems addressed since the previous IPR);
- A review of the T&E results, including an analysis of the T&E results made by one or more additional qualified experts;
- A review of any problem uncovered by the IPR to be handled as described under unscheduled IPR.

Possible outcomes of the scheduled IPR are:

- To accept the ES prototype as meeting its goals.
  - If this is not the last iteration, then proceed to the next iteration.
  - If this is the last iteration then move into final testing and the Milestone 2 review.

  - Terminate the project.
- To accept the ES prototype as a completed ES even though its goals were not met.
  - If this is not the last iteration, then change the relevant Demonstration Prototype plans and T&E plans and proceed to the next iteration.
  - If this is the last iteration, then declare the current ES to be the finalized ES prototype and move into final testing and the Milestone 2 review.
  - Terminate the project.

The IPR proceedings and outcome should be reported in the Progress Notebook.

6.4.4 Finalized Prototype Evaluation Experiment (T3)

A final T&E of the Demonstration Prototype should be performed immediately following the final IPR and preceding the Milestone 2 review.

This T&E test consists of ES-trained but project-naive end-users using the demonstration ES to solve test cases. The task will be conducted in the same way as the T&E task held at the end of each iteration except that this evaluation will be made by end-users not previously associated with the project. The reason for this evaluation is to obtain an unbiased end-user view of the ease/difficulty of use, understandability, adequacy of coverage, and capability of the ES.

The T&E team will monitor this activity including interviewing the end-users about any problems and suggestions for improvements. If at all possible, these sessions should be recorded using audiovisual equipment. Such recordings will be very valuable in developing the Testbed Prototype. The results of this T&E activity should be documented in the Progress Notebook.

Problems exposed by the evaluation should be written up in problem reports and documented in the Progress Notebook. All problems should be reviewed by the T&E team, KE, and PM to determine their seriousness and how they should be handled. If the project is ending, then nothing should be done other than documenting the problems. If it is assumed that the project will continue after the Milestone 2 review, then any decision as to how to handle the problems depends on a risk analysis.

Since the Demonstration Prototype serves as a design tool for the Testbed Prototype and will not be used to maintain the deployed system, it may be decided that documenting the problems and their solutions rather than implementing the solutions may be a cost-effective option (i.e., the risk in eliminating the implementation is less than the time and cost required to do the implementation). If the risk is too great, then another Demonstration Prototype iteration should be planned, funded, and performed.

6.4.5 Application Checklist Update (T4)

The finalized Demonstration Prototype provides a complete description of the application space with respect to the ES, an initial design, and an implementation. On the basis of these, a much more accurate assessment of development needs can be made. The application checklist should be updated to reflect changes as a result of developing the Demonstration Prototype as well as changes that have occurred external to the project, such as to relevant AIS(s), I/O devices, security, and telecommunications.

In reference to the example checklist offered in Section 5.4.3, the Demonstration Prototype design should enable refinements to:

- Application performance needs,
- ES performance-enhancing techniques,
- KBS representational power,
- End-user interface support needs,
- Developer-user interface support needs,
- Knowledge acquisition support needs,
- Configuration management needs,
- Test and evaluation needs,

- Maintenance needs,
- Automated documentation needs,
- Life-cycle development needs.

6.4.6 Architecture Update (T5)

Changes in AIS architecture, special devices, security, and telecommunications could affect the other two categories listed in Section 5.4.3: integration needs, and security and integrity needs. If changes in these areas imply an architecture change, conceptual prototyping should be used to demonstrate a new solution. The concept prototype may be: a description of an existing production solution that will be emulated; a design and walk through; a limited demonstration using simulated data and system interfaces; or a real demonstration. The PM must decide when the architecture should change. The PM may want to address these changes at the time he/she first becomes aware of them rather than waiting until this point in the process.

New priority profiles and critical characteristics lists should be made for the Testbed Prototype effort and the deployed system. A new specification for the ES tool to be used in testbed prototype development and deployment should be made based on the new profiles.

The updated application checklist, profiles, and ES tool specification should be documented in the Progress Report.

6.4.7 Acquisition Strategy Update (J2)

This is a joint technical and management activity. The reader is referred to Section 5.4.5 for a discussion about developing an acquisition strategy.

Two acquisition strategy decisions need to be reconsidered at this time: (1) the validity of an earlier specification for the Testbed Prototype tool (if one was made) given the more current requirement definition; and (2) the acquisition plan for carrying out the Testbed Prototype effort.

If the new testbed profile is considerably different from the previous one, and if the ES tool for the Testbed Prototype was selected as part of the earlier SOW and is currently under acquisition, then a new evaluation should be made to find out if a different tool would yield a better technical solution. If so, a cost/benefit analysis should be performed to determine whether to use the earlier tool or to re-specify the Testbed Prototype tool as part of a new SOW. The new requirements that may particularly affect tool choice are: performance in terms of speed required, size of the ES, user interface support, and maintenance (a rapidly changing knowledge base may require a tool with good developer support functions).

The PM may wish to alter the earlier acquisition strategy as a result of his/her experience in developing the Demonstration Prototype. For example, if the project has been conducted in-house but it has been difficult to obtain good staff and keep them, the PM may want to contract out the rest of the project; or if the contractor responsible for the Demonstration Prototype did an exceptionally good job, the PM may want to continue with the contractor and his existing staff rather than go for another competitive bid.

6.4.8 The SOW for Testbed Prototype (T6)

After the Milestone 2 Management Review has been held and the project has been given the go-ahead, an SOW should be written whether the project will be supported in-house or implemented under contract.

The SOW should include:

- The functional definition and design of the problem taken from the Progress Notebook, FD and SS sections, and developed for the Demonstration Prototype to include:
  - The scope of the problem to be addressed in the Testbed Prototype and, if applicable, a broader scope with indications as to when the additional breadth/depth will be provided (e.g., post-deployment follow-ons). Include detailed description of the Demonstration Prototype architecture including

knowledge bases and databases, all important architecture decisions and reasons for decisions, and outstanding issues and problems.
  - Tool specifications including the application checklists and profiles for the Testbed Prototype and deployed ES.
  - Environment description: current and future AIS integration, embedding, special device interfaces, security requirements, and data requirements. Include descriptions and results from the conceptual prototypes.
  - Information about the user community (availability of experts, end users, and test data), about the maintenance community, and about the availability of government ADP and security personnel to aid in definition/design.
  - Discussion of all risk factors and open issues, including the results of risk analysis and recommendations for resolving the risks.
- Example plan for development of the Testbed Prototype, including:
  - Plan for the entire development effort, including level of effort and schedule, staff requirements, and scheduled IPRs.
  - Description of the desired finalized Testbed Prototype based on the finalized Demonstration Prototype (include videotape if possible).
  - Plan for each testbed iteration, including functional description and estimated cost and schedule.
  - Business plan to show capability to incrementally fund and modify schedule, and plans to accommodate refinements as a result of IPRs.

If the Testbed Prototype is to be competitively bid, then the following should be included in the RFP:

- Qualification requirements of the contractor, including:
  - Expert-system development experience by project.
    - Description of project, including current state of effort.
    - Project size in terms of ES tool used.
    - Project size in terms of staff, person-years, and cost.
    - Client contact person.
  - System integration experience by project.
    - Description of project, including current state of effort.
    - Project size in terms of person-years and cost.
    - Client contact person.
  - Relevant domain experience.
    - Projects performed.
    - Experience of identified personnel.
  - Experience levels with specific ES tools and AI machines and an indication of those in-house and available for use.
  - Size and experience of the ES staff.
  - Identification of key personnel available to work on the project.
- Intended scope of SOW. SOW will cover the Testbed Prototype effort only or a phased program of which the Testbed Prototype effort will be the first phase. If phased program, indicate how acquisition of additional phases will be handled.

7. DEFINITION/DESIGN PHASE, TESTBED PROTOTYPE[5]

7.1 INTRODUCTION

Figure 7.1 shows the place in the expert system development process of the Definition/Design Phase Testbed Prototype. Its major technical and management activities are shown in Figure 7.2. The gray boxes there indicate the process for handling architecture changes. The Definition/Design Phase consists of two prototype efforts, the Demonstration Prototype described in Chapter 6 and the Testbed Prototype described here. This phase is concerned with producing a complete design and re-implementation of the problem definition exemplified by the Demonstration Prototype. The result will be a stand-alone prototype ES. The testbed implementation may be accomplished by (1) using the ES tool (or a family member of the tool) that will be used in the deployed system or (2) coding the ES directly in a DoD-acceptable programming language. The testbed effort should be of lower risk than the Demonstration Prototype effort. (This effort may also be of lower risk than the Operational Prototype effort where integration considerations, although carefully considered, still pose major uncertainties.) The problem has been well-defined and its solution illustrated by the Demonstration Prototype. There should be no major surprises concerning the problem solution.

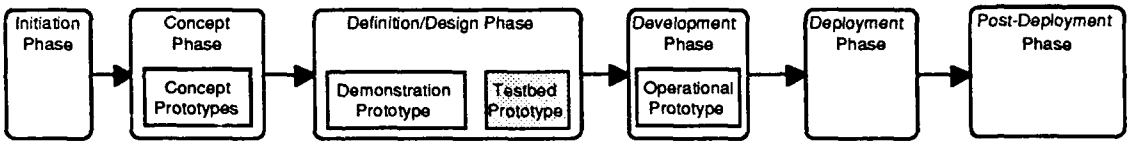


Figure 7.1—Expert system development process, Testbed Prototype

Although ES tool selection for the Demonstration Prototype stressed the flexibility of the tool in supporting design, tool selection for the Testbed Prototype first stresses compatibility with the deployed system and, only second, flexibility of the tool in supporting design. In the ideal situation, the Demonstration and Testbed Prototypes would be built using the same ES tool with flexible design support, and a runtime version of the tool, streamlined for performance, would be used in the Operational Prototype and deployed system. For some tools, the switch from development to runtime may be accomplished by simply changing parameter settings; in the most extreme case the development and runtime tools are implemented in different languages (e.g., development tool in LISP, runtime version in C) and/or the underlying rule/knowledge base language is different.

Decisions about what version of a tool to use will need to be made on the basis of the tool characteristics and the application needs. For example, if fast performance of the stand-alone ES is a high priority, the PM may decide that the Testbed Prototype should be developed using a runtime ES tool in order to evaluate performance of the prototype .

In any case, the Demonstration Prototype will be an aid in building the Testbed Prototype and then will be discarded. All life-cycle maintenance will be accomplished by changes to and T&E of the Testbed and/or Operational Prototypes. It is important that the ES tool used in the testbed development provide sufficient latitude to accommodate the expected and unavoidable minor changes in the fielded expert system. However, though the

[5]See Appendix A for definitions of acronyms and terms used in this chapter.

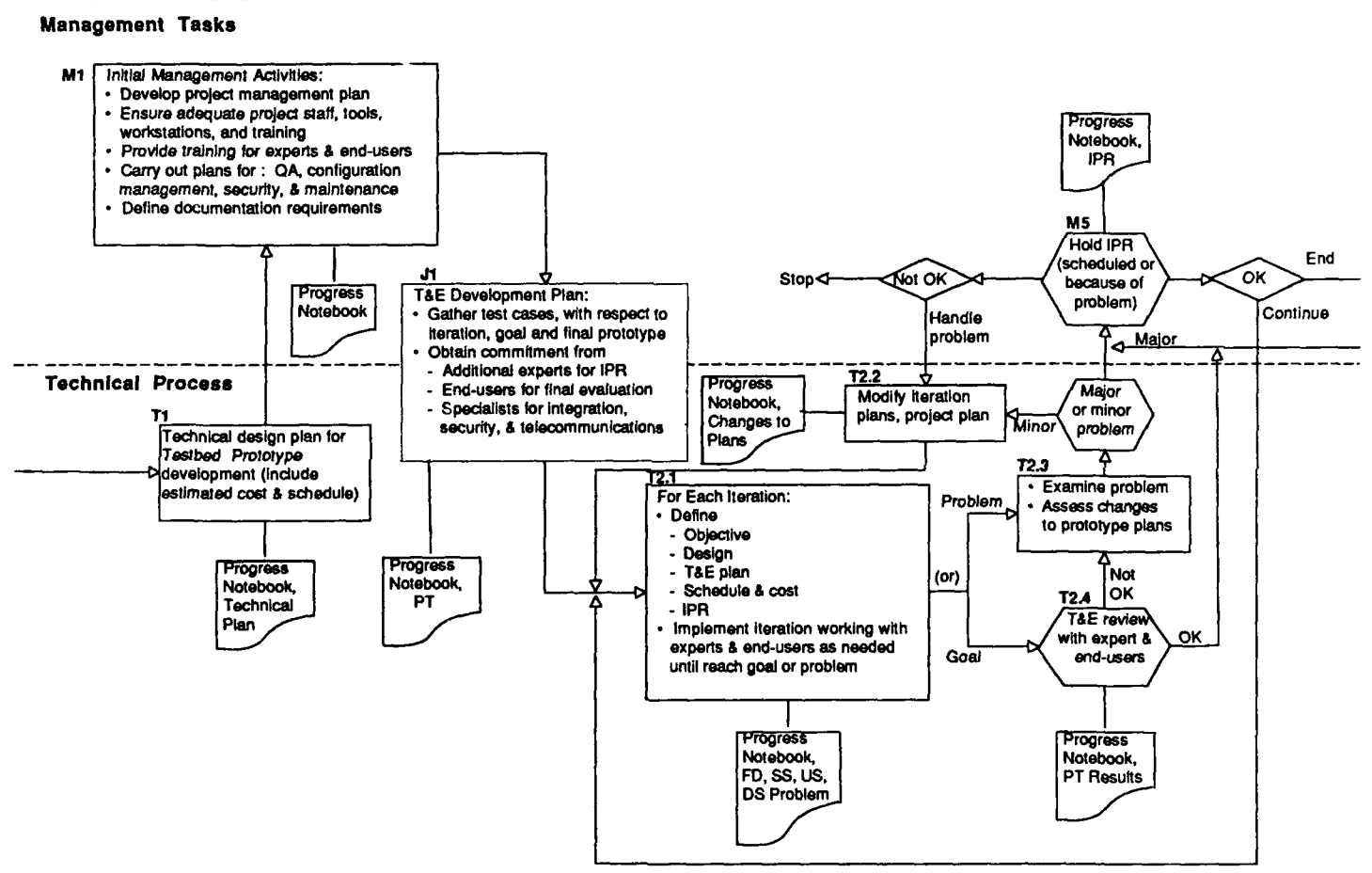
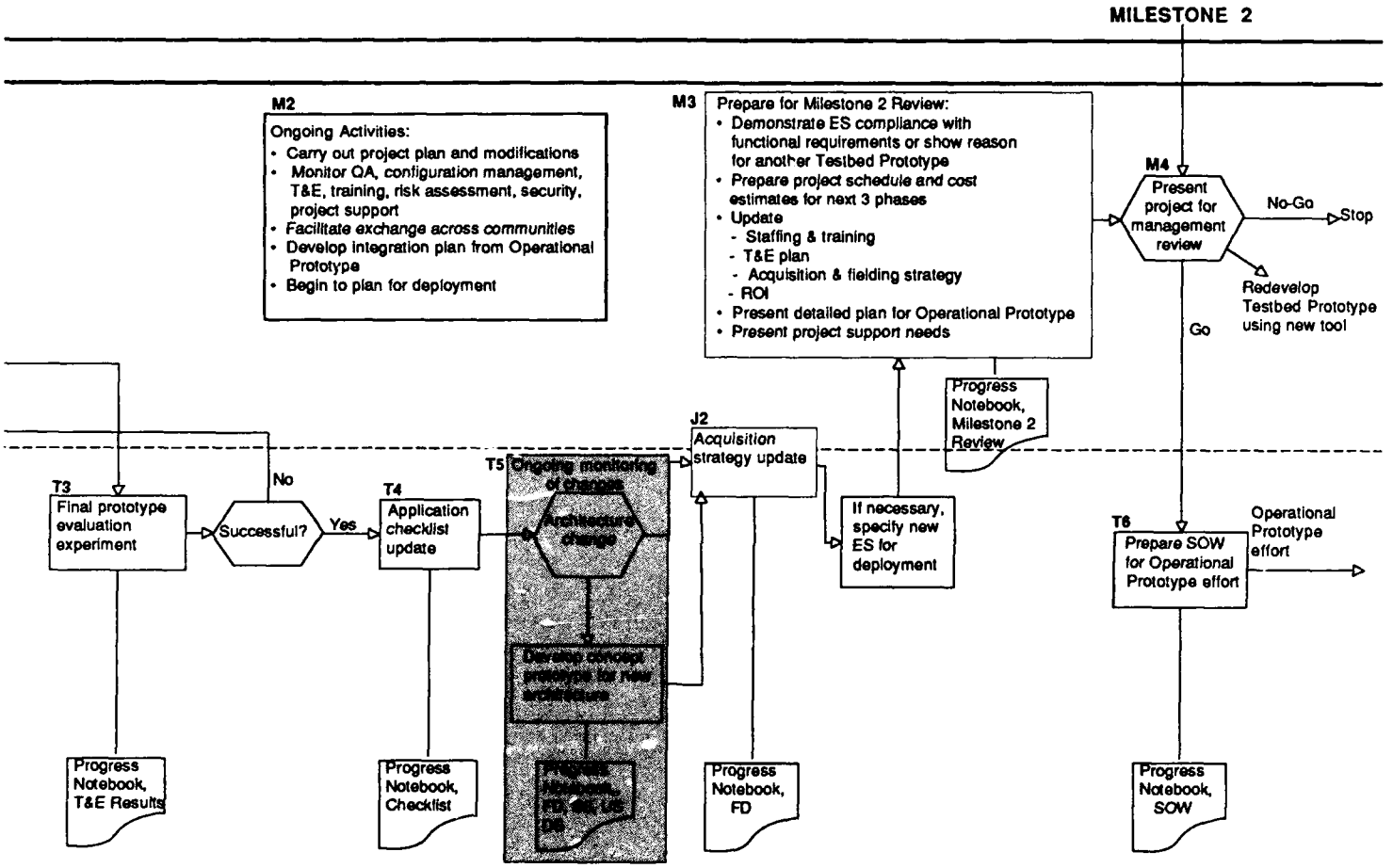


Figure 7.2—Definition/Design Phase, Testbed Prototype


MILESTONE 2
M2
Ongoing Activities:
• Carry out project plan and modifications
• Monitor QA, configuration management, T&E, training, risk assessment, security, project support
• Facilitate exchange across communities
• Develop integration plan from Operational Prototype
• Begin to plan for deployment
M3
Prepare for Milestone 2 Review:
• Demonstrate ES compliance with functional requirements or show reason for another Testbed Prototype
• Prepare project schedule and cost estimates for next 3 phases
• Update
- Staffing & training
- T&E plan
- Acquisition & fielding strategy
- ROI
• Present detailed plan for Operational Prototype
• Present project support needs
M4
Present project for management review
No-Go
Stop
Redevelop Testbed Prototype using new tool
Go
Progress Notebook, Milestone 2 Review
J2
Acquisition strategy update
T5
Ongoing monitoring of changes
Architecture change
Develop develop prototype for new architecture
Progress Notebook, FD, etc. to DB
T3
Final prototype evaluation experiment
Successful?
No
Yes
T4
Application checklist update
If necessary, specify new ES for deployment
T6
Prepare SOW for Operational Prototype effort
Operational Prototype effort
Progress Notebook, T&E Results
Progress Notebook, Checklist
Progress Notebook, FD
Progress Notebook, SOW

Demonstration Prototype is discarded, all the problems, solutions, descriptions, documentation, issues, and decisions that occurred during its design and development will still be recorded in the Progress Notebook for ready reference during the product life cycle. For example, issues that resulted in reducing the scope of the problem early on should be revisited in the Post-Deployment Phase when product upgrades are being considered to extend the scope.

The testbed effort begins after the acquisition strategy decided upon and implemented in the Demonstration Prototype effort has resulted in an in-house commitment or contractual choice to carry out the SOW.

Overview of Management Activities. Initial management activities (M1) include: developing a management plan; ensuring that the project is adequately staffed and provided with the necessary ES tools, workstations, and training; ensuring necessary expert and end-user training; carrying out plans developed for QA, configuration management, security, and maintenance; and defining the documentation requirements for the Testbed Prototype effort. After this, the PM and technical staff will jointly define a detailed T&E plan for the Testbed Prototype.

Ongoing activities (M2), also performed during the Demonstration Prototype effort, include: carrying out the project plan and modifications made to it as a result of the prototyping effort; monitoring QA, configuration management, T&E, training, risks (changes to AIS(s), I/O devices, etc.), and security and project support; and facilitating exchanges across the developer, user, maintenance, ADP, and security communities. New ongoing activities are to develop the integration plan for the Operational Prototype including initial site selection and to begin planning for deployment (e.g., installation sites, order of installations, training, and maintenance).

The PM will conduct in-process reviews (IPRs) at the conclusion of each prototype iteration or on demand as major unanticipated problems arise (M5). The final Milestone 2 review (M4) will: demonstrate ES compliance with the functional requirements; present the project schedule and cost estimation for the next three phases; present an update on staffing and training, T&E plans, acquisition and fielding strategy, and ROI; present the detailed plan for carrying out the Operational Prototype Phase (including end-user training and maintenance); and present project support needs. The result of the final Milestone 2 review is: to continue the project by moving on to development of the Operational Prototype; to iterate back into the Testbed Prototype activity because the ES was found to be inadequate in meeting the functional requirements; or to terminate the project.

Overview of the Technical Process. The technical process begins by defining a detailed technical design and plan for developing the Testbed Prototype (T1). The ES will be defined in terms of its overall functionality, cost, and schedule, then a detailed design will be developed followed by a breakdown into iterations. The next step is a joint technical/management preparation of the T&E plan (J1) to specifically define the test database, and obtain commitments from additional experts to review the prototype at IPRs, and obtain commitments of selected end-users to perform the final evaluation of the Testbed Prototype.

The iterative development of the Testbed Prototype is next (T2). Each iteration has a well-defined objective, design, T&E plan, schedule, cost, and IPR. Though not expected, any major problem arising during an iteration will trigger an unscheduled IPR. Each IPR (M5) will determine if the project is proceeding well in meeting its plan, if the plans should be changed, if the testbed is completed, or if the project should be terminated. During development of the Testbed Prototype the KE will use the expert more for general review and for consulting when problems arise, rather than working as closely with the expert as he/she did during Demonstration Prototype development. T&E results will be reviewed by the expert and end-users. IPRs will be conducted as they were in the previous efforts—experts and end-users will participate and the T&E results will be reviewed and critiqued by additional expert(s).

At the end of the prototype development, the final T&E (T3) will include the experimental use and evaluation of the Testbed Prototype by a small group of end-users who have not been involved in the prototype effort. The application checklist will be updated (T4) to reflect changes to the problem definition, AIS architecture, I/O devices, and security and telecommunication needs. Architecture changes (T5) external to the application should have been monitored for and detected when they occurred. If necessary, new Concept Prototypes should have been built to show how integration, interfacing, security, and communication changes will be handled in the Operational Prototype. This effort, if needed, should proceed as much as possible in parallel with development of the stand-alone ES prototypes.

Changes in the architecture may necessitate re-evaluation of the acquisition and fielding strategy (J2) with regard to interface devices, I/O devices, communication networks, etc., for the Operational Prototype and for the deployed systems. All activities undertaken during the testbed prototyping effort that are of current or may be of future value should be documented in the Progress Notebook and where appropriate under the headings FD, SS, US, and DS. The Progress Notebook will be an input to the management preparation for the final Milestone 2 review. The last technical task is to write an SOW for the Operational Prototype (T6).

This chapter is divided into four sections: Section 7.1, this section, is an introduction; Section 7.2 presents an example risk containment table; Section 7.3 describes the management activities; and Section 7.4 describes the technical process.

7.2 RISK CONTAINMENT

Table 7.1 is an example prioritized list of major risk items that need to be identified and addressed during the Testbed Prototype effort. Since applications vary, the project manager should develop his/her own list of the top ten risk items.

7.3 MANAGEMENT ACTIVITIES

The management activities are organized into:

7.3.1 Initial Management Activities (M1)
7.3.2 Development of T&E Plan for Testbed Prototype (J1)
7.3.3 Ongoing Management Activities (M2)
7.3.4 Acquisition Strategy Update (J2)
7.3.5 Preparing for the Milestone 2 Management Review (M3)
7.3.6 Holding the Final Milestone 2 Management Review (M4)

7.3.1 Initial Management Activities (M1)

The initial management activities are: (1) to develop a management plan; (2) to ensure that the project is adequately staffed and provided with the necessary ES tools, workstations, and training; (3) to ensure necessary training for expert(s) and end-users; (4) to carry out plans developed for QA, configuration management, security, and maintenance; and (5) to define the documentation requirements for the Testbed Prototype effort.

Management Plan. A management plan should be prepared that includes: technical plan milestones; schedule for acquiring staff; schedule for acquiring ES tools and equipment; training schedule (for staff, experts, and end-users); documentation schedule; maintenance schedule; IPR schedule; and the schedule of monthly meetings to review progress and risk management.

Staffing, Equipment, and Training. The technical staff requirements were addressed in the SOW and should be reviewed with the KE or technical manager and modified if necessary. Staffing, equipment, and training may not be an issue if the same ES tool and

Table 7.1

AN EXAMPLE PRIORITIZED LIST OF TESTBED PROTOTYPE RISK ITEMS

	Risk Item	Risk Management Techniques
1.	Personnel shortfalls	Staff this phase with top talent; ensure that tools and equipment are available; provide training
2.	Lack of community participation	Obtain commitment for: qualified expert(s) to work with KE; additional experts for review of ES; sufficient number of end-users; test cases with criteria; integration, security, and telecommunications specialists as needed
3.	Changes to relevant AIS(s), I/O devices, security, and telecommunications	Detect external system changes early on; evaluate changes; refine concept prototype if necessary to demonstrate new architecture
4.	Inadequate performance (speed or size)	Test performance speed and size by simulation; build concept prototype on runtime tool
5.	Gold plating	Avoid adding additional functionality; document and save for post-deployment upgrades
6.	Unrealistic schedule and expectations	Be realistic in plan and schedule; do careful design and decomposition into incremental prototypes; recognize when cost and schedule are inadequate for design; recognize and handle problems when they occur
7.	Inefficiency and costliness of monthly reviews and unscheduled IPRs	Hold informal monthly reviews with little paper work; ensure quick unscheduled IPRs with minimum formal preparation, restrict attendees to PM's discretion
8.	Documentation: maintaining the Progress Notebook	Assign project librarian to be responsible for maintaining all documentation; define how different types of information not in FD, SS, US, DS, and PT will be documented in Progress Notebook; oversee and review documentation; ensure documentation is validated
9.	Modification of existing acquisition strategy	Recognize requirement changes and project experiences that may affect changes to acquisition strategy; perform cost/benefit analysis of changing the strategy
10.	Effect of changes in requirements on validity of problem to user mission	Evaluate mission needs whenever considering major changes to problem, integration, security, or telecommunications; evaluate mission need if redoing Testbed Prototype

workstations are being used for the Testbed Prototype as were used for the Demonstration Prototype and if most of the Demonstration Prototype staff will be carrying out the effort. If a different ES tool is being used, then existing staff may have to be trained in using the ES tool or new staff with such experience may have to be acquired. Additional staff may also be required because of any schedule constraint. Staff acquisition and training are of prime

concern, and the technical schedule should reflect a realistic estimate of when trained staff will be available to work. If the work is being done in-house and staff is unavailable, the PM may want to consider hiring qualified consultants. Expert system staffing is discussed in Section 5.3.3.5 and training in Section 5.3.3.8.

The timely acquisition of ES tools and environment (e.g., workstations) is also critical. If the technical staff is being trained locally, it is very important for them to have the tools available during that time.

Expert and End-User Training. If the ES tool being used is different from the one used for developing the Demonstration Prototype, provide additional training to expert(s) and end-users. Types of training necessary for experts and end-users are discussed in Section 5.3.3.8. End-users who participated in the Demonstration Prototype effort should also participate in the Testbed Prototype development. In addition, if a site has been selected for the Operational Prototype development, it may be desirable to have several end-users from that site participate in the training and the Testbed Prototype development effort.

Carrying out Plans. It is the PM's responsibility to see that plans developed for QA, configuration management, security, and maintenance are carried out.

- The PM should ensure that new technical staff have a good understanding of QA guidelines and configuration management procedures.
- If a different ES tool and/or workstation will be used, the configuration management support tools may be different from those used previously. The PM should ensure that the project librarian has special training and/or computer aids in order to best carry out the configuration management functions.
- If the Testbed Prototype is a classified project, the PM must carry out the security plan with regard to: physical environment (including workstations and disks); clearances for staff members, experts, users, ES/workstation maintenance personnel (if required), and others as required; labeling of inputs and outputs; user/developer access and authorization procedures; etc. If the ES tool and/or workstations have changed, new security procedures may be required.
- The PM must plan for ES tool/workstation installation and checkout; ongoing maintenance of equipment; and either vendor or contractual support for the ES tool or a resident tool smith.

Documentation Requirements. The PM should define the documentation requirements for the testbed prototyping effort, make them known to the technical staff, and document them in the Progress Notebook. The minimum amount of documentation is shown in Figure 7.2 and was derived from Figure 3.2, "Automated information systems life-cycle documentation," taken from Draft DoD Standard 7935.1, 15 January 1988. The documentation requirements should identify specific sections in the Progress Notebook for the Functional Description (FD), the System/Subsystem Specification (SS), the Software Unit Specification (US), Database Specification (DS), Test Plan (PT), and the Operational Prototype SOW. The PM should designate further sections of the Progress Notebook for information, issues, and problems not provided for in the above-mentioned sections. Further sections to be considered are: project management plan; project technical plan; T&E test results; "problem" descriptions and resolutions; application checklist; and reports on IPRs and monthly management meetings.

The PM should designate a project librarian who will be responsible for maintaining the documentation.

7.3.2 Development of T&E Plan for Testbed Prototype (J1)

Developing the T&E plan for the Testbed Prototype is a joint technical and management activity. It is addressed in Section 7.4.2 as part of the technical process.

7.3.3 Ongoing Management Activities (M2)

Ongoing management activities also performed during the Demonstration Prototype phase include: carrying out the project plan and modifications made to it as a result of the prototyping effort; monitoring QA, configuration management, T&E, training, risks (e.g., changes to AIS(s), I/O devices, etc.), security, and project support; and facilitating exchanges across the developer, user, maintenance, ADP, and security communities.

New ongoing activities are to develop the integration plan for the Operational Prototype, including initial site selection, and to begin planning for deployment (e.g., installation sites, order of installations, training, and maintenance).

Managing the Project Plan. The objective of the Testbed Prototype is to design and re-implement the Demonstration Prototype using some version of the same ES tool that will be used in the deployed system. The Demonstration Prototype has answered in the affirmative the question, "*Can* the problem be solved functionally?" and the Demonstration Prototype itself is an example of a solution. The Testbed Prototype is addressing the question, "*How should* the problem solution be designed and implemented in order to provide the functionality of the Demonstration Prototype and also meet the performance and maintenance requirements of the deployed system?" The PM must ensure that the technical staff addresses the performance requirements. They may do this, for example, by devising a realistic way of emulating data flows and component timings; or, if the ES tool being used is only a version of the deployed tool, they may develop limited prototypes using the runtime ES system to test critical performance or memory-intensive sections of the program.

Although most major issues should have been addressed by the demonstration prototyping effort, issues that may arise in this effort include: changes to the earlier design functionality, and trade-offs (e.g., modularity makes maintenance easier but may penalize speed of performance). A recognized risk at this effort is "gold plating"—adding functionality to satisfy an additional need or a broader user community. This should be avoided whenever possible. However, gold plating suggestions should be documented so they will be available in the Post-Deployment Phase to consider as part of the product upgrade.

To accommodate unforeseen problems and issues, the testbed prototyping effort must be able to easily and inexpensively facilitate changes to the technical and management plans, costs, and schedules. This requires careful management oversight of the process so that deviations from established goals, schedules, and costs are quickly recognized and addressed without imposing an excess of management demands on the technical team. The two review techniques for achieving this are: informal monthly project reviews, and more formal, in-process reviews (IPRs).

The monthly reviews should concentrate on progress, risk management, and problems that occurred during the previous month. The review could be in the form of a short written report, a discussion between the PM and KE and perhaps a user organization representative, or a meeting of all or selected members of the project staff.

The IPR (M5) can be either scheduled (at the end of each prototype development iteration—approximately 6-month intervals) or unscheduled (called because of a major problem). The scheduled IPR addresses whether the ES has met its goals, the T&E results, problems to be resolved, risk management, revisions to plans, and the next step (e.g., continue the prototype, go on to the next iteration, conclude the project). Attendees may include representatives from all interested communities and should, as a minimum, include the project staff, the user organization manager, and experts and end-users participating in the project development.

The unscheduled IPR is held to address a major problem; attendance should be decided by the PM with recommendations from the KE or technical leader.

Activities that Need Monitoring. In addition to monthly reviews and IPRs, the PM should:

- Direct the KE or technical leader in monitoring QA.

- Monitor the configuration management activity and address any management problems that occur (e.g., technical staff evading the process).
- Monitor the T&E activity addressing any problems with the plan, acquisition of test cases, participation of the user community, and follow-up on technical response to results.
- Monitor all training activities by talking with people who have been trained and their managers to determine that the training is being done well and is sufficient. If not, the PM should look into making other arrangements.
- Monitor risks. Though these will be addressed in monthly meetings, some risk management should be assumed directly by the PM. The most critical of these, in this effort of development, are risks external to the stand-alone ES prototype, such as changes to an AIS or I/O device with which the deployed system will be integrated or changes to security or telecommunication requirements. If such a change affects the overall architecture, the PM should see that the technical team addresses the architecture change as soon as possible and develops a new design and concept prototype.
- Monitor security policy and methods to ensure that they are being properly enforced. New methods or ways of enforcing security may be needed either because of a change in the ES tool and/or workstation or because the Testbed Prototype is the first step toward the operational system and accreditation in the lab is needed—the Demonstration Prototype may have had fewer restrictions than accreditation in the field will have.
- Monitor project support needs: maintenance of tools and equipment, office and laboratory space and usage, secretarial and documentation support, etc.

Facilitate Exchange Across Communities. The PM should facilitate information exchange across user, developer, ADP, security, and maintenance communities by keeping them informed and inviting them to IPRs. In particular, the PM should establish close ties with the user community and frequently discuss user satisfaction with the user organization manager and with experts and users who are participating in the project. The PM has a responsibility to keep the user organization manager informed of project needs for expert and end-user participation, including training and T&E. The project cannot succeed unless a good relationship is established between the user and developer organizations and unless the user organization accepts responsibility for its part of the project effort.

The PM should begin to plan for integration by selecting the integration site, by arranging for I/O devices to connect to, by selecting a supply of "real" data for testing, and by ensuring the participation of the user, security, systems, and maintenance communities. Initial plans or considerations for deployment should begin during this phase. This should include order of installations, training of users and maintenance personnel, and acceptance testing.

As the Testbed Prototype develops, the PM should brief the project to higher management to attract champions and generate enthusiasm and support. If there are plans to deploy the ES to users at different sites, this is a good time to begin to brief them on the project. As mentioned in the Concept Phase, use of a videotape presentation to demonstrate the ES prototype is highly recommended.

7.3.4 Acquisition Strategy Update (J2)

Updating the acquisition strategy so that it effectively integrates the technical, business, and management elements of the project is a joint technical and management effort. It is discussed in Section 7.4.7 as part of the technical process.

7.3.5 Preparing for the Milestone 2 Management Review (M3)

The main purpose of the Milestone 2 review is to ascertain that the Testbed Prototype (1) fulfills the functional requirements of the application as a stand-alone solution and (2) is

valid in meeting the mission needs of the user organization. This also includes showing that the timing and performance requirements of the integrated solution, to be implemented by the Operational Prototype, are highly likely to be met.

Information to be presented at the review includes:

- Testbed Prototype compliance with the functional requirements, including security.
- Project plan, schedule, and cost estimation for the next three phases.
- An update on:
  - Test and evaluation plans.
  - Staffing and training.
  - Acquisition and fielding strategy.
  - Return on investment.
- Detailed plan for carrying out the Operational Prototype phase (including end-user training and maintenance).

Discussion of Testbed Prototype Compliance with Functional Requirements. The PM should prepare a high-level walk through the design and show how it meets the functional requirements presented in the Functional Description (FD). Discuss any changes in the problem scope since the first Milestone 2 review and, if changes have occurred, the validity of the current FD to the user community's mission. Discuss the risks avoided with regard to straining ES capabilities and gold plating, and discuss problems that have arisen and their solutions. Discuss all changes (actual, planned, anticipated) in AIS and/or I/O devices (to be integrated with the ES), security, and telecommunications that necessitate changes to the ES architecture. Present the new concept prototype architecture that addresses these changes and explain effects of the new architecture on the overall plan. If the deployed ES will be a secure system, address how the Testbed Prototype satisfies the security requirements and what (if any) changes will need to be made for the Operational Prototype and the deployed ES system. A short videotape presentation of the final Testbed Prototype may be an effective way to demonstrate the problem solution.

In the case where a cost/benefit analysis has shown that the effort would be best served by selecting a different deployment ES tool and recycling through the testbed prototyping effort again, present the case for another Testbed Prototype cycle. This should include a discussion of how it will affect the user mission and a discussion of all preparations for the first Milestone 2 review covered in Section 6.3.5 at the end of the Demonstration Prototype effort.

Discussion of Project Schedule and Cost Estimation for the Next Three Phases. Schedules and cost estimates for the remaining phases should be developed. They should be derived from information gathered about:

- The size and level of effort required for the integration activities.
- The size of electronic and text source data and knowledge and an estimate of the effort needed to acquire them for the Operational Prototype and for the deployed system.
- Estimated amount of effort and cost for: maintenance during Operational Prototype development, deployment, and post deployment (including the KBA maintaining the ES and the changing knowledge base); training of Operational Prototype staff (including ADP, security, and telecommunications personnel), users, experts, and maintenance personnel; T&E; configuration management; and documentation.
- Estimated cost of project management activities and reviews.

Again, as in the previous cost estimates, the costs of other similar efforts should be compared with the new cost estimate to ensure that it is not off by an order of magnitude or more.

Update of T&E Plans. Any changes to the T&E plans should be presented. Experience in developing the Testbed Prototype may indicate a need for better criteria and for changes in the number and kind of test cases. Selection of a site to be used in development of the

Operational Prototype may influence the selection of end-users and experts needed for the next phase.

Update of Staffing and Training Needs. Experience in developing the Testbed Prototype and in planning for the Operational Prototype should result in a more accurate assessment of the type and number of staff required.

Update of Acquisition and Fielding Strategy. An update to the acquisition and fielding strategy should accommodate any changes in the plans to acquire hardware and software for the Operational Prototype development and deployment. This will include any special software/hardware needed to interface to AIS(s) or telecommunications networks as well as changes in special I/O devices. Advantages to ordering in large batches should be discussed, along with risk (e.g., if the requirements change before deployment), availability of the equipment, vendor support, maintenance, and training. At this point in the development process, a fairly detailed plan for deployment should have been developed. The acquisition and fielding strategy and the plan must be consistent.

Update of the Cost Benefit/ROI. The PM should refine the cost benefit/ROI presented at the Milestone 2 review based on changes made to the FD during the testbed prototyping. Examples of changes in the problem definition that may affect cost benefit/ROI are: narrowing or expanding the problem scope; and changing the overall system design due to changes in relevant AIS(s), I/O devices, security, and telecommunications. In addition, there may be an increase or decrease in the estimated user community need for the ES product.

Discussion of Detailed Plan for Carrying out the Operational Prototype Phase. Describe the detailed plan that will be included in the SOW for the Operational Prototype. It should include a description of the site, users, integration functions to be designed and integrated, and a phased plan for doing so. Included will be: additional technical staff needs such as ADP, security, and telecommunications specialists; additional user participation needs such as site resident end-users and experts; additional maintenance staff and their needs for any special equipment; relevant training for all people involved; the development of a training program for the users of the deployed system; and additional documentation requirements, which will include users' manual, end-user manual, maintenance manual, and implementation procedures.

7.3.6 Holding the Milestone 2 Management Review (M4)

The last management activity of the Testbed Prototype effort is to present the project at the Milestone 2 Management Review. Managers at all levels of the participating organizations should be encouraged to attend, as well as expert and end-user representatives from the user organization and members of the ADP, security, telecommunications, and maintenance organizations. As mentioned earlier, a videotape may be an effective way of describing the finalized Testbed Prototype.

The result of the final Milestone 2 review will be: to continue the project by moving on to development of the Operational Prototype; to iterate back into the Testbed Prototype activity because the ES was found to be inadequate in meeting the functional requirements; or to terminate the project.

7.4 TECHNICAL PROCESS

The technical process of the Testbed Prototype effort can be organized into the following tasks:

- 7.4.1 Technical Design and Plan for Testbed Prototype Development (T1)
- 7.4.2 Development of T&E Plan for Testbed Prototype (J1)
- 7.4.3 Testbed Prototype and Iterations (T2)
- 7.4.4 Finalized Prototype Evaluation Experiment (T3)
- 7.4.5 Application Checklist Update (T4)

7.4.6 Architecture Update (T5)
7.4.7 Acquisition Strategy Update (J2)
7.4.8 The SOW for Operational Prototype (T6)

7.4.1 Technical Design and Plan for Testbed Prototype Development (T1)

The Testbed Prototype begins with the definition of the problem and the design and implementation of the problem solution provided by the Demonstration Prototype (which are described in the SOW and the Progress Notebook, particularly under the Functional Description (FD) and System/Subsystem Specification (SS)). The accuracy and completeness of the problem definition as demonstrated by the prototype have been accepted by the user organization as meeting the functionality of their needs and has passed management review as well. However, the design and implementation of the prototype was a side effect of defining the problem. Rapid prototyping was used to quickly demonstrate the solution. The resulting prototype is most likely very breakable, the code may be kludged (though it started out modular, adding new functions and fixing old ones probably has resulted in a difficult-to-maintain program), and performance and size issues were not necessarily addressed (an exception may be a Demonstration Prototype of a real-time ES).

The Testbed Prototype is a clean start and may be built from a different ES tool than the Demonstration Prototype. The problem definition, experience in building the Demonstration Prototype, and the Demonstration Prototype itself are used in developing a design of the Testbed Prototype. This design is driven by several goals: fulfilling the problem definition, providing program modularity for ease of understanding and maintainability, and meeting the user organization performance requirements. It may happen that these goals will conflict and trade-offs will need to be considered during the prototyping process.

The top-level Testbed Prototype design will show the general functionality to be supplied by major components and subcomponents (or modules). Internal interfaces between components and modules will be identified. Since the Testbed Prototype is a stand-alone ES, components or modules will be assigned the functions of emulating external interfaces to AIS(s) and/or special I/O devices. If the operational ES is intended to be embedded within an AIS, then the user interface component of the Testbed Prototype should emulate the user interface that will be furnished by the AIS. Once the top-level design is complete, an overall technical plan can be laid out that consists of the functional description of the problem solution, the design, cost, and schedule.

In developing the Demonstration Prototype it was suggested that (1) the problem solution be prioritized in terms of the categories to be handled and (2) the priorities be used to determine what to prototype first. The objective was to be able to quickly demonstrate important parts of the solution in order to sustain user interest and build support. The testbed prototyping effort has a different agenda. The breakdown of the work into iterations should make sense in terms of the overall design rather than by priority.

The scope of the problem to be addressed by each iterative prototype should (1) make sense in terms of the overall design, (2) build on the ES of the previous iteration, (3) consist of enough of the problem solution to produce a demonstrable ES that is meaningful to users who will be evaluating it, and (4) be able to be developed in approximately 6 months. A module in an early iteration may be developed to the degree that is needed to support the iteration goal, or it may be fully developed in accordance with the overall design. Any partial implementation of a module should be consistent with the design for the full implementation. Functionality may be added to a module in later iterations, but the module should not have to be redesigned.

The plan for each iteration should include the functional description of the resulting ES and its design, cost, and schedule. Cost and schedule of the iteration should be determined after the functional description and detailed design are determined. The testbed prototyping process incorporates the fact that changes to the technical plan may occur. With adequate

time for design, however, problems are expected to occur much less frequently than in the demonstration prototyping process, with less impact on cost and schedule.

The resulting cost and schedule for the testbed prototyping effort should be realistic. If the cost and/or schedule are fixed and inadequate to meet the design requirements, then the technical leader should so indicate to the PM, and the issue should be resolved before any implementation takes place. If the problem definition has to be reduced in scope in order to meet the cost and schedule constraints, then the newly scoped problem should be re-evaluated to be sure it still meets the user organization mission needs. If it does not, then the project may be terminated. The objective is to realistically and precisely define the end conditions of the prototyping process within the cost and schedule constraints.

The technical plan will be documented in the Progress Notebook.

7.4.2 Development of T&E Plan for Testbed Prototype (J1)

As soon as the technical plan is completed, the T&E team with the help of the PM should develop detailed T&E plans. This includes (1) acquiring the necessary database of test cases and determining the cases and criteria to be used to test the results of each prototype iteration, (2) committing to additional qualified expert(s) to participate in IPRs, (3) selecting end-users to participate in the final evaluation experiment, and (4) if appropriate, selecting specialists in security, ADP, and telecommunications to verify that the design and Concept Prototypes will meet the integration, security, and telecommunication requirements for the next phase.

Test Cases. Sufficient test cases may have been gathered under the initial T&E plan to supply the Testbed Prototype development effort. If so, the staff should determine if the problem definition and scope have changed enough to require additional test cases or invalidate some of the existing ones. Any additional needs should be conveyed to the user organization manager.

The number and structure of test cases is dependent on the application. However, the user community should be made aware of the importance of (1) providing test cases that cover the problem space and (2) furnishing enough of these to allow the development team and the T&E team to add fresh test cases for each prototype iteration. Unlike a conventional software system (e.g., a payroll package) where there is a "correct" answer for each test case, an ES response may be rather a response that is considered adequate and reasonable from the expert's point of view (reasonableness of the answer may be determined from the ES explanation). Test cases are the best way to test the coverage of the ES and how prone it is to failing at the edges of the problem space. In general, the more varied the test cases are, the better the ES will cover the problem space and demonstrate resilience at the edges—and it will be more likely to fail gracefully rather than to break down.

Test cases should be evaluated by experts, independently of the ES development, to decide what the desired response or set of possible responses to each case should be. If possible, each response should also include its rationale. These responses and their rationales are the criteria against which the ES will be judged. Agreement should be reached about how the ES responses will be rated against the criteria. If the rationales are not included with the criteria, then the experts evaluating the system will have to judge whether the ES explanation for a response is acceptable. If the test cases are real cases that have been collected with their actual responses, then comparison of the actual responses with the ES responses would be very instructive.

Additional Qualified Experts. The value of the ES will depend on how successfully it captures, uses, and explains expert knowledge. The objective of T&E is to test the ES by evaluating its responses against the test case criteria. The validity of the criteria is beyond T&E—it is the responsibility of the user community, especially the domain experts. Additional qualified experts should be made available to the project specifically to review and critique the T&E results presented at each IPR, and possibly to help in determining the test case criteria. In Chapter 4 we mentioned the importance of the expert's credibility. It is

equally important that additional experts be recognized as credible and qualified authorities. The validity of the ES depends very much on the validity of the experts.

If the site for the Operational Prototype has been selected, and none of the current participating experts comes from that site, then it may be desirable to get one or more qualified experts from that site to participate in the IPRs. This would allow site experts to validate the Testbed Prototype before the operational prototyping phase and get them educated, interested, and involved.

Final Evaluation Experiment. The final T&E test will be an evaluation experiment in which end-users unfamiliar with the project will receive end-user training and a set of test cases to solve using the ES. If the Operational Prototype site is known, it would be beneficial to have end-users from that site participate in the final T&E. User performance should be recorded in detail (audiovisual equipment would be very useful here) and evaluated by the T&E team, experts, and the project team. In order to carry out the experiment: evaluation criteria must be defined, end-user participation guaranteed by the user organization, end-user training scheduled, the experiment scheduled, and analysis planned for. The results should feed into improvements to the user interface implemented either by an additional final iteration of the Testbed Prototype or as part of the operational prototyping effort.

Final Evaluation of Integration, Security, and Telecommunications. If appropriate, specialists in ADP, security, and telecommunications should evaluate how well those needs have been addressed in the Concept and/or Testbed Prototypes. The objective is to achieve a consensus that relevant issues have been addressed and that the risk is low in proceeding to the Operational Prototype phase.

Detailed T&E Plan. A detailed T&E plan should be prepared showing schedule, participants, and amount of effort involved in: preparing test cases and criteria; running T&E prior to scheduled IPRs; and carrying out the final evaluation experiment. A version of the plan should be prepared for the manager of the user organization that will show the schedule of user participation required as part of the T&E process. Lack of sufficient user participation is a reflection of end-user interest. Insufficient end-user interest may warrant cessation of project activities.

The T&E plan, user organization plan, test cases and criteria, and problems and issues should be documented in the Progress Notebook IP section.

7.4.3 Testbed Prototype and Iterations (T2)

General requirements for each Testbed Prototype iteration are:

- A well-defined prototype plan, including a functional description of the prototype iteration, performance requirements, design, cost, and schedule;
- An expert(s) available to the KE(s) for consulting and review as needed;
- Trained end-users available for T&E at the end of a prototype iteration;
- The test cases that will be used for development, each with a description of the functionality it is testing, performance characteristics (if appropriate), the correct result, and the rationale supporting that result;
- A planned IPR using additional qualified experts to evaluate the T&E results.

Steps in the process of developing a prototype iteration are:

7.4.3.1 Plan for Prototype Iteration (T2.1)
7.4.3.2 Iteration Development (T2.2)
7.4.3.3 Handling Problems (T2.3)
7.4.3.4 Test and Evaluation of Iteration (T2.4)
7.4.3.5 In-Process Review of Iteration (M5)

7.4.3.1 Plan for Prototype Iteration (T2.1)

Each Testbed Prototype iteration is an ES development effort with a well-defined goal, design, cost, and schedule.

The prototype plan will include a functional description of the top-level ES architecture and its decomposition into components and modules, performance requirements (if appropriate), the test cases and criteria to be used in developing the ES, work assignments, and estimated schedule and cost to carry out the development.

The top-level architecture will describe the broad functions to be performed and the assignment of those functions to components and modules. Internal interfaces between components and modules will be identified; external interfaces will be emulated in components or modules. Each component and module should be identified as to whether it is to be: acquired intact from an earlier iteration; modified from an earlier iteration; or developed as a new ES element.

Work assignments and schedule will indicate who or which teams will be responsible for development of the various elements.

7.4.3.2 Iteration Development (T2.2)

The technical leader should be sure all technical personnel are aware of the QA requirements and should periodically review code to be sure the QA requirements are adhered to. For large projects, it is especially important to establish how the teams will coordinate their efforts when using the established configuration management practices.

Iteration development will be done according to the design, plan, and schedule. Several issues that may need to be addressed by the developers are discussed below:

- Avoiding the risks in straining ES technology
- Selecting of programming and representation methods
- Refining the user interface
- Refining data and knowledge needs
- Testing performance requirements
- Handling "problems"
- Determining the end of the iteration

Avoiding the Risks in Straining ES Technology. The developers should refrain from implementing a technical solution that strains ES technology. If there is no other solution, and the need has not been previously addressed, then the developer should describe this as a "problem" in a problem report and it should be handled accordingly.

Selecting Programming and Representation Methods. The design should indicate the programming and representation methods to be used (as a result of the concept and demonstration prototyping explorations), but a developer may have insight showing that a different method would be better. If so, this should be discussed with the technical leader and, if he/she concurs, be written up in a "problem" report. If there is no impact on the cost and schedule, then the PM should be informed, the change to the design made, and implementation continued. If there is an impact on cost and schedule, then the PM and technical leader will decide how to handle the change (i.e., forgo it, or hold an unscheduled IPR).

Refining the User Interface. The user interface requirements should be well specified, but if some modifications were required to the Testbed Prototype (e.g., as a result of the final user evaluation), the KE may need to interact with end-users before the IPR. The KE should try to keep end-user and expert involvement at a minimum during this period. However, if a real need exists and the KE determines that a risk may be incurred by not increasing the expert and/or end-user involvement, he/she should make the need known to the PM and user organization and try to get the needed assistance.

Refining Data and Knowledge Needs. Interactions with experts and users may uncover additional or different data and knowledge needs. If so, the KE should translate the needs into specific data/knowledge requirements and document these in the Progress Notebook under the Functional Description (FD) and Database Specification (DS) sections. Any newly identified data/knowledge source or substantial change in the amount to be acquired and

maintained should be recognized as a "problem," and a problem description should be written and made known to the technical leader and PM.

Testing Performance Requirements. Performance requirements should be tested according to the design. This may necessitate emulating data flows; testing a limited prototype using the runtime ES; obtaining ADP help to get various readings on AIS data flows, telecommunications flows, etc.; emulating an embedded system user interface; etc. These tests should be carefully designed, implemented, run, and documented in the Progress Notebook.

7.4.3.3 Handling Problems (T2.3)

A problem report should be prepared (and documented in the Progress Notebook) for any problem whose solution will substantially affect the plan. (This includes schedule and cost deviations.) The report will consist of: a definition of the problem, alternative solutions, and their respective effects on the current iteration plan and on the finalized Testbed Prototype.

The KE and PM should decide when a problem is minor and can be handled by an internal change to the iteration and T&E plans. If so, the change should be made and documented in the Progress Notebook, and the iteration should continue.

If the KE and PM decide the problem is major, then an unscheduled IPR should be held, at which time the problem will be presented for adjudication.

Determining the End of the Iteration. Since deviation from cost and schedule is handled as a problem, the iteration will be ended when the iteration goal is met (the planned functional description is realized according to design) or as a result of an unscheduled IPR.

7.4.3.4 Test and Evaluation of Iteration (T2.4)

Test and evaluation will be held at the end of the iteration. Both expert(s) and end-users will participate in T&E. There will be four sets of tests:

- Test cases from previous T&E runs (including Demonstration Prototype T&E runs) will be rerun to establish the fact that the new ES supports the established T&E baseline. It is possible that a major change will induce some test cases to fail (e.g., if a category has been removed). Any failures should be individually addressed by the KE and expert.
- New test cases will be run by the T&E team (with the help of a technical team member if necessary) and the responses recorded and analyzed with respect to the criteria. Special tests may need to be run to demonstrate the ability of the ES to meet the performance requirements.
- New test cases will be run by the expert and responses recorded and analyzed by the expert with respect to the criteria. If the rationale has not been provided, then the expert should judge how well the ES handled the reasoning behind the response.
- End-users will use the system to handle selected test cases with technical staff help if necessary. Their performances will be monitored and reported by the T&E staff. After using the system, the end-users will be interviewed about the problems they had and their suggestions for improvements. If possible, the end-user tests should be recorded using audiovisual equipment.

The T&E session should be reported in the Progress Notebook.

A problem report should be prepared for any recognized problem and documented in the Progress Notebook. The T&E leader, KE, and PM should decide if a problem is minor and can be handled by an internal change to the iteration plan and a minor change to the T&E plan (which includes rerunning the T&E after the changes have been made). If so, the changes to the various plans should be made and documented in the Progress Notebook and the iteration should continue.

If the T&E leader, KE, and PM decide the problem is major, then the problem should be addressed at the scheduled IPR, which will be held immediately after the T&E report is prepared.

7.4.3.5 In-Process Review of Iteration (M5)

An IPR can be either scheduled or unscheduled. A scheduled IPR should be held immediately after the T&E to review the status of the project and, in particular, to evaluate the current ES prototype. An unscheduled IPR is called when a major problem is identified that may cause considerable change to the plans and schedule. Unscheduled IPRs may be called during a Testbed Prototype iteration or as a result of the final evaluation test.

The people who should attend the IPR include: all project staff and managers, the user-community project expert and additional expert(s), the end-users who participated in the pertinent development effort and T&E, and user organization managers. The PM should invite representatives from the ADP, security, and maintenance communities who are involved in testing performance requirements for the Testbed Prototype or those who will be involved in the Operational Prototype.

Unscheduled IPR. The unscheduled IPR should address the current status of the project in meeting its goals and objectives, the problem that prompted the IPR, alternatives for its solution, and recommendations from the technical team. The PM (with recommendations from the KE) should decide whom to invite to the unscheduled IPR.

What to do about the problem will depend on a number of factors:

- How well the project is going.
- How important the problem solution is with respect to the worth of the application to the user community and its mission if the problem is not solved. If it is worth solving then:
  - Availability of additional funding,
  - Acceptability of lengthening the schedule and/or finding additional staff.
- The amount of risk involved in the solution with respect to straining technology (ES, ADP, security, telecommunications, etc.).
- The degree of "gold plating" (capabilities outside the defined problem scope) required to solve the problem.

Alternative outcomes of the IPR are:

- To rescope the application to eliminate the need to solve the problem, to change relevant plans, and to continue the ES prototype.
- To obtain additional funding and a schedule extension and/or additional staff to solve the problem, to change relevant plans, and to continue the ES prototype.
- To declare the current ES iteration to be the finalized prototype. Stop the Testbed Prototype effort at this point, and perform the end-of-iteration T&E, the final testing and other activities, and the final Milestone 2 review.
- To declare the current ES iteration to be the finalized prototype and terminate the project.

The IPR proceedings and outcome should be reported in the Progress Notebook.

Scheduled IPR. The scheduled IPR should include:

- A review of the project's progress to date and how well it is meeting its objectives (this should include reference to all major and minor problems addressed since the previous IPR);
- A review of the T&E results, including an analysis of the T&E results made by one or more additional qualified experts;
- A review of any problem uncovered by the IPR to be handled as described under "Unscheduled IPR."

Possible outcomes of the scheduled IPR are:

- To accept the ES prototype as meeting its goals.
  - If this is not the last iteration, then proceed to the next iteration.
  - If this is the last iteration then move into final testing and the final Milestone 2 review.
  - Discontinue the project.
- To accept the ES prototype as a completed ES even though its goals were not met.
  - If this is not the last iteration, then change the relevant Testbed Prototype plans and T&E plans and proceed to the next iteration.
  - If this is the last iteration, then declare the current ES to be the finalized ES prototype and move into final testing and the final Milestone 2 review.
  - Discontinue the project.

The IPR proceedings and outcome should be reported in the Progress Notebook.

7.4.4 Finalized Prototype Evaluation Experiment (T3)

A final T&E of the Testbed Prototype should be performed immediately following the final IPR and preceding the Milestone 2 review.

This T&E consists of ES-trained (but project-naive) end-users using the testbed ES to solve test cases. The task will be conducted in the same way as the T&E task held at the end of each iteration except that this evaluation will be made by end-users not previously associated with the project. If the site for the Operational Prototype has been selected, it is suggested that these end-users be from that site. The reason for this evaluation is to obtain an unbiased end-user view of the ease/difficulty of use, understandability, adequacy of coverage, and capability of the ES.

The T&E team will monitor this activity including interviewing the end-users about any problems and suggestions for improvements. If at all possible, these sessions should be recorded using audiovisual equipment. Such recordings will be valuable to the rest of the phases. The results of this T&E activity should be documented in the Progress Notebook.

Problems exposed by the evaluation should be written up in problem reports and documented in the Progress Notebook. All problems should be reviewed by the T&E team, KE, and PM to determine their seriousness and how they should be handled. If the project is ending, then nothing should be done other than documenting the problems. If it is assumed that the project will continue after the final Milestone 2 review, then any decision on how to handle the problems depends on a risk analysis.

If the identified problems are high risk because of the technical difficulty of their solution, then it may be decided to implement their solutions in an additional testbed iteration. If the identified problems are low risk, then it may be decided to implement their solutions during the Operational Prototype effort. If the latter is decided, then the problems and their solutions should be documented in the Progress Notebook.

7.4.5 Application Checklist Update (T4)

The finalized Testbed Prototype provides a complete description of the application space with respect to the ES, an initial design, and an implementation. On the basis of these, a much more accurate assessment of the remaining development needs can be made. The application checklist should be updated to reflect insights gained from the testbed prototyping as well as changes that have occurred external to the project, such as changes to relevant AIS(s), I/O devices, security, and telecommunications.

In reference to the example checklist offered in Section 5.4.3, the Testbed Prototype design should enable refinements to:

- Application performance needs,
- ES performance-enhancing techniques,
- KBS representational power,

- End-user interface support needs,
- Developer-user interface support needs,
- Knowledge acquisition support needs,
- Configuration management needs,
- Test and evaluation needs,
- Maintenance needs,
- Automated documentation needs,
- Life-cycle development needs.

7.4.6 Architecture Update (T5)

Changes in AIS architecture, special devices, security, and telecommunications could affect the other two categories listed in Section 5.4.3: integration needs, and security and integrity needs. These areas should be continuously monitored by the PM during this effort of development. When a significant change is detected, the PM should request that it be addressed immediately rather than waiting until the final T&E. The technical staff with the help of appropriate specialists should develop a concept prototype to demonstrate a new solution. The concept prototype may be: a description of an existing production solution that will be emulated; a design and walk through; a limited demonstration using simulated data and system interfaces; or a real demonstration.

A new priority profile should be made for the deployed system and compared with the old profile to see if there are any major differences that would cause problems in proceeding with the deployment plans. Examples of differences might be the discovery: that the ES had to accommodate a much larger database or knowledge base than previously planned, as such size would overtax the memory available in the deployed tool; or that the number of potential end-users had grown significantly and the cost of the deployed ES tool and workstation was too high so a less expensive tool and workstation should be considered. If the changes are significant and call for a change in deployment tool and thus a new Testbed Prototype effort, a specification should be written for the new ES tool.

The updated application checklist, profiles, and ES tool specification should be documented in the Progress Report.

7.4.7 Acquisition Strategy Update (J2)

This is a joint technical and management activity. The reader is referred to Section 5.4.5 for a discussion about developing an acquisition strategy.

The acquisition strategy may need to be updated because of either: (1) the need to change the deployment ES and develop a new Testbed Prototype before proceeding further, or (2) the need to change equipment and/or the way in which the Operational Prototype and deployment will be carried out.

If the new deployment profile is considerably different from the previous one, then an evaluation and tool selection should be made to find out if a different selection would yield a better technical solution. If so, a life-cycle cost/benefit analysis should be performed to determine, for example, whether going forward using the more expensive ES tool for deployment will cost more than selecting a less expensive ES and repeating the Testbed Prototype effort. Other examples will involve different trade-offs, such as proceeding with an ES tool that will quickly degrade in performance because of memory constraints or proceeding with an ES tool that the vendor will not be supporting.

The PM may need to alter the earlier acquisition strategy as a result of new information about deployment plans. For example: more sites want to use the ES than previously planned; the security accreditation requirements may have changed and workstations may have to be specially engineered or the Operational Prototype may have to be developed in a secure environment; or special interfacing equipment may be needed because of changes in the telecommunication rates or because of new networks.

The PM may also want to change the acquisition strategy as a result of his/her experience in developing the Testbed Prototype. For example, if the Testbed Prototype effort

was contracted out, the PM may or may not want the same contractor to continue with the next phase depending on the contractor's performance, or there may be reasons (such as security) for preferring to do the integration in-house.

7.4.8 The SOW for Operational Prototype (T6)

After the Milestone 2 Management Review has been held and the project has been given the go-ahead, an SOW should be written whether the project will be supported in-house or implemented under contract.

The SOW should include:

- The functional definition and design of the problem taken from the Progress Notebook–Functional Description (FD), System/Subsystem Specification (SS), Software Unit Specification (US), Database Specification (DS), and other sections developed for the Testbed Prototype to include:
  - The scope of the problem to be addressed in the Operational Prototype and, if applicable, a broader scope with indications as to when the additional breadth/depth will be provided (e.g., post-deployment follow-ons).
  - A detailed description of the Testbed Prototype architecture emphasizing the modules that emulate the interfaces to be developed and including knowledge bases and databases, all important architecture decisions and reasons for decisions, and outstanding issues and problems.
  - Deployment tool choice and the application checklist and profile for the deployed system.
  - Detailed environment description including: all AISs that will be integrated with, interfaced to, embedded within; each special interfacing I/O device; security architecture requirements; and telecommunication requirements and use of specified networks and devices. The performance requirements should be precisely specified. A description of the Concept Prototype's architecture should be described in detail along with results of all T&E and other evaluations.
  - Information about the user community (availability of experts, end-users, and test data), about the maintenance community, and about the government ADP, security, and telecommunications specialists to aid in development.
  - Discussion of all risk factors and open issues, including the results of risk analysis and recommendations for resolving the risks.
- Example plan for development of the Operational Prototype, including:
  - Plan for the entire development effort, including level of effort and schedule, staff requirements, and scheduled IPRs.
  - Description of the desired finalized Operational Prototype based on the finalized Testbed Prototype and concept prototype of integration (include videotape if possible).
  - Plan for each operational prototype iteration, including functional description and estimated cost and schedule.
  - Business plan to show capability to incrementally fund and modify schedule, and plans to accommodate refinements as a result of IPRs.

If the Operational Prototype is to be competitively bid, then the following should be included in the RFP:

- Qualification requirements of the contractor, including:
  - Expert-system development experience by project, indicating in more detail those projects using the specified ES tool and requiring similar integration.
    - Description of project, including current state of effort and integration ease/difficulty.
    - Project size in terms of ES tool used.

    - Project size in terms of staff, person-years, and cost.
    - Client contact person.
  - System integration experience with respect to specific AIS(s), I/O devices, security, and telecommunications needs stated in preceding bullet, including:
    - Description of project, including current state of effort.
    - Project size in terms of person-years and cost.
    - Identification of key personnel available to work on the project.
    - Client contact person.
  - Relevant domain experience, including:
    - Projects performed.
    - Experience of identified personnel.
  - Experience levels with the ES tool being used and an indication of whether the tool is in-house and available for use.
  - Size and experience of the ES staff.
  - Identification of key personnel available to work on the project.
- Intended scope of SOW. SOW will cover the Operational Prototype effort only or a phased program where the next phase is deployment. If phased program, indicate how acquisition of the Deployment and Post-Deployment Phases will be handled.

8. DEVELOPMENT PHASE, OPERATIONAL PROTOTYPE[6]

8.1 INTRODUCTION

Figure 8.1 shows the place in the expert system development process of the Development Phase Operational Prototype. Its major technical and management activities are shown in Figure 8.2. The gray area in that figure indicates the iterative development of the Operational Prototype.

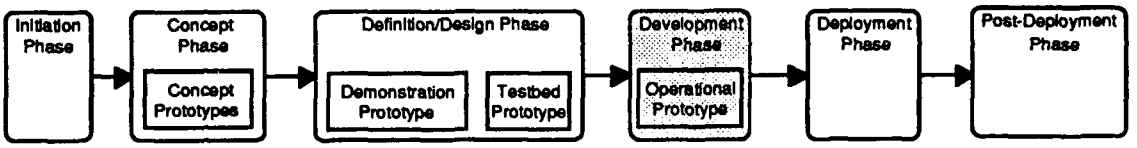


Figure 8.1—Expert system development process, Operational Prototype

The Operational Prototype objective is to integrate the stand-alone Testbed Prototype into an example operational environment using the actual deployed ES tool, AIS(s), special I/O devices, and telecommunications networks. The end result will be a prototype product that demonstrates complete integration capability and meets performance and system security requirements. Performance requirements include speed and correctness as well as recoverability and robustness. If the ES is to be used continuously for many hours or non-stop 24 hours a day, it must be demonstrated that it can maintain itself (e.g., garbage collection of old data and knowledge) without degradation over long periods of time. Recoverability from various types of hardware and software errors must also be demonstrated (e.g., pulling the plug on the networks, the disks, the ES workstations).

The final Operational Prototype should use the runtime version (as opposed to the development version) of the deployed ES tool if available. However, earlier prototype iterations may use a development version if one is available. For some ES tools the switch from development version to runtime may be as simple as changing parameter settings. If so, the technical team may switch between the two versions, using the development version as an aid in debugging interfaces and the runtime version for testing the performance characteristics of the system. If the development version is used during operational prototyping, then the Operational Prototype rather than the Testbed Prototype may be used for making all ES modifications now and for prototyping ES modifications in this phase and on into its entire life cycle. The runtime version should not be used due to its lack of development-oriented diagnostic aids.

The schedule and cost estimates for this phase, unlike for the stand-alone Testbed Prototype phase, may be high-risk items because integration within the operational environment involves many unknowns. For example, LoPiccolo [1988] quotes Philippe Brou of Ascent Technologies: "When I price systems, I place connectivity at 60 percent of the total cost. Connectivity between different computers and different types of equipment is worse than communicating with people in foreign languages." Brou also says that even if a vendor claims its product is compatible with a standard communications protocol such as TCP/IP, connectivity is not straightforward. If someone has to install a different controller on the machine, the whole program may have to be rewritten.

[6]See Appendix A for definitions of acronyms and terms used in this chapter.

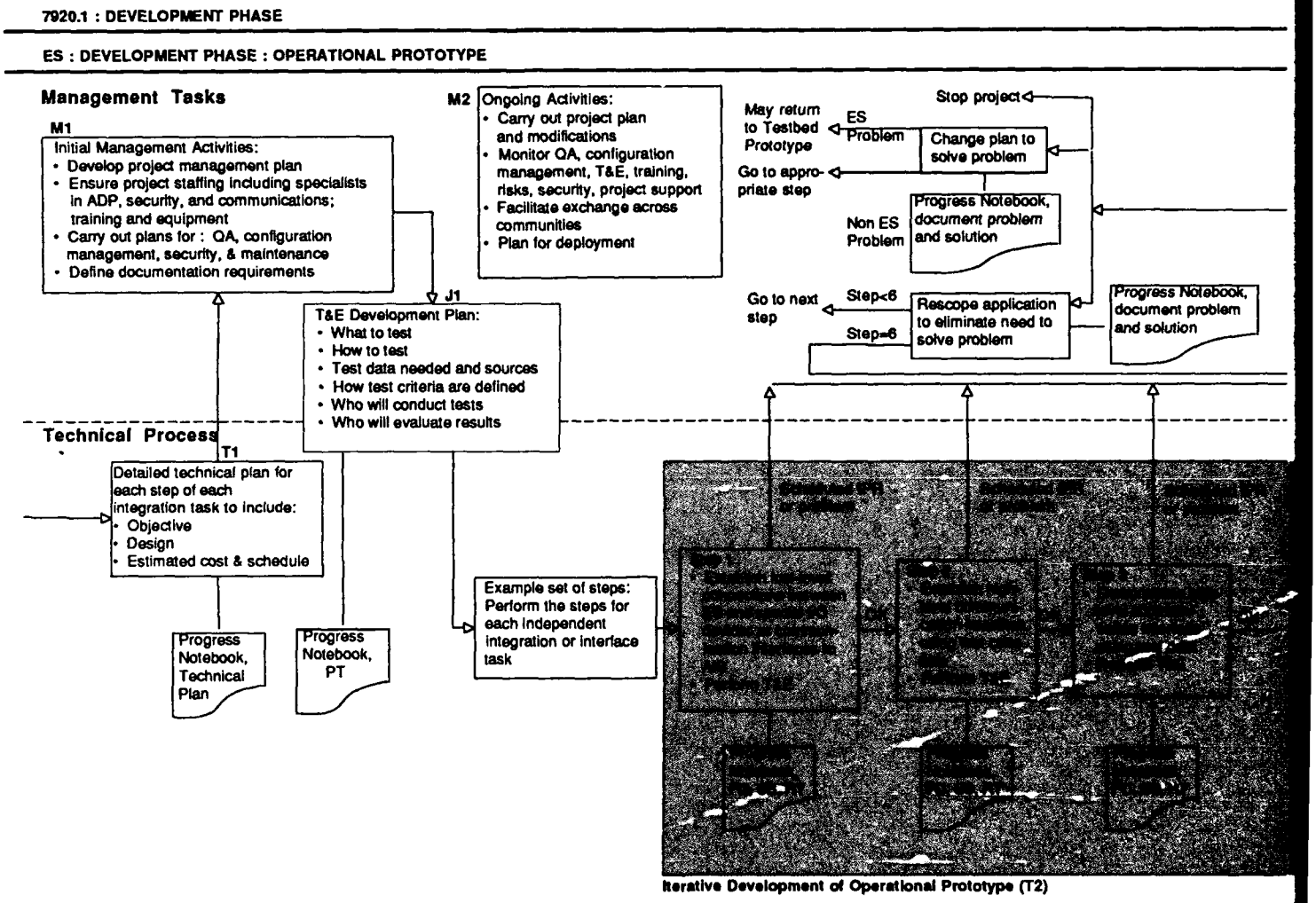
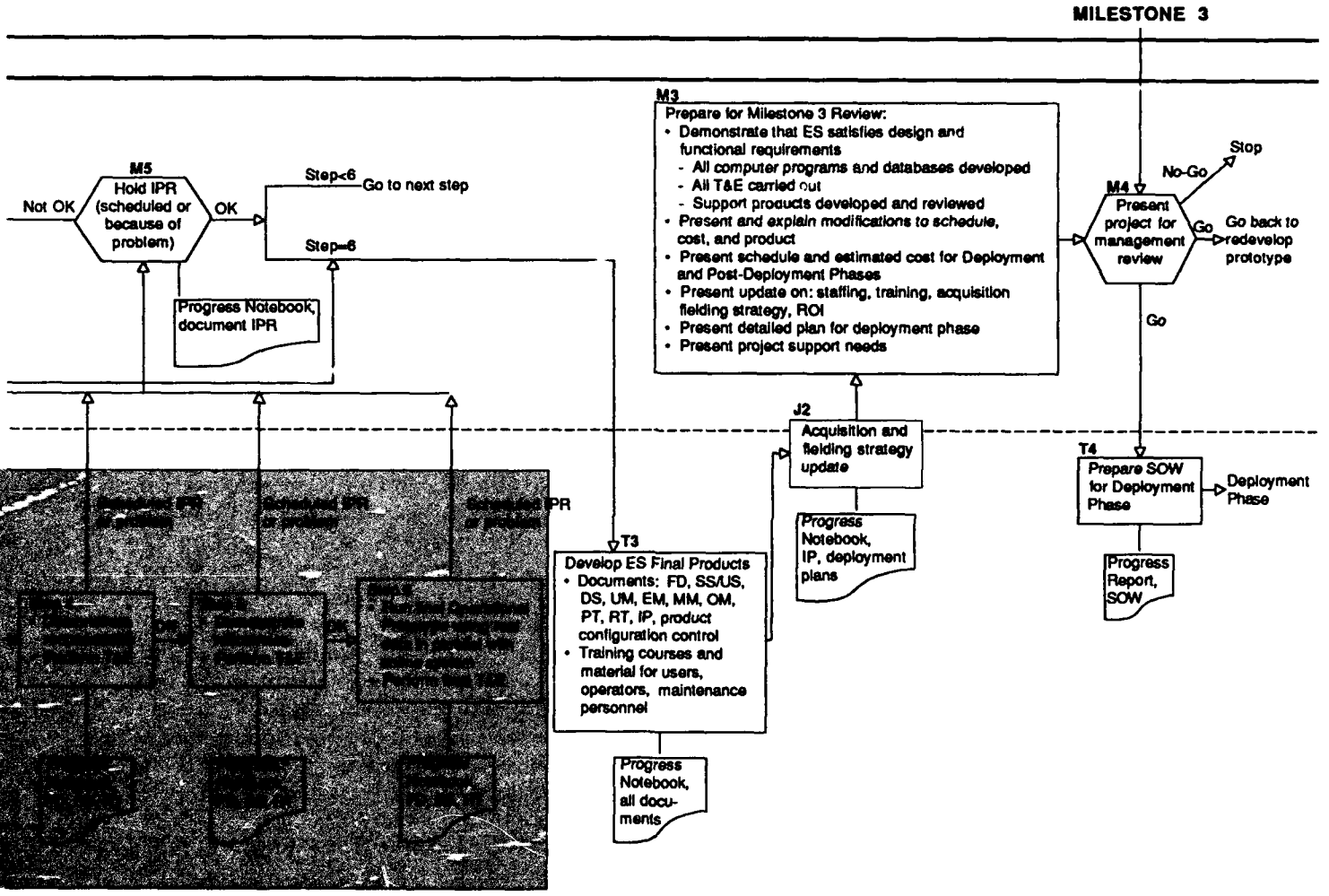


Figure 8.2—Development Phase, Operational Prototype


MILESTONE 3
M3
Prepare for Milestone 3 Review:
• Demonstrate that ES satisfies design and functional requirements
- All computer programs and databases developed
- All T&E carried out
- Support products developed and reviewed
• Present and explain modifications to schedule, cost, and product
• Present schedule and estimated cost for Deployment and Post-Deployment Phases
• Present update on: staffing, training, acquisition fielding strategy, ROI
• Present detailed plan for deployment phase
• Present project support needs
M4
Present project for management review
Stop
No-Go
Go
Go back to redevelop prototype
Go
M5
Hold IPR (scheduled or because of problem)
Not OK
OK
Step<6
Go to next step
Step=6
Progress Notebook, document IPR
J2
Acquisition and fielding strategy update
Progress Notebook, IP, deployment plans
T3
Develop ES Final Products
• Documents: FD, SS/US, DS, UM, EM, MM, OM, PT, RT, IP, product configuration control
• Training courses and material for users, operators, maintenance personnel
Progress Notebook, all documents
T4
Prepare SOW for Deployment Phase
Deployment Phase
Progress Report, SOW

Schoen and Sykes [Schoen 1987, page 168] say, "Deferring a capability for on-line operation till later does not mean that providing on-line operation can be done immediately or is inexpensive. In fact, it is likely that the integration of an AI system may involve considerably more effort and cost than has been expended in the development of the original prototype. A preliminary evaluation of expected integration issues should be made prior to a major commitment to specific hardware or software or both and a reasonable reserve for cost growth and schedule slip provided."

Paying close attention to integration and interface needs from the beginning is a risk-reduction approach but is still no guarantee against surprises. The importance of updating Concept Prototypes whenever the interfacing or integrating externals change cannot be overemphasized.

Overview of Management Activities. Initial management activities (M1) include: developing a management plan; ensuring that the project is adequately staffed with relevant specialists in ADP, security, communications, and maintenance as well as ES technology; providing staff, expert, and end-user training; ensuring that needed equipment is available and supported; carrying out plans developed for QA, configuration management, security, and maintenance; and defining the documentation requirements for the Operational Prototype effort.

Ongoing activities (M2) include: carrying out the project plan and modifications made to it as a result of the prototyping effort; monitoring QA, configuration management, T&E, training, risks, security of the Operational Prototype environment, and project support; facilitating exchanges across the developer, user, maintenance, ADP, security, and telecommunications communities; and planning for deployment in terms of sites, order of installations, acquisition of hardware and software, and training of end-users, operators, and maintenance personnel.

The PM will conduct in-process reviews (IPRs) (M5) at approximately 6-month intervals or on demand as major unanticipated problems arise. The Milestone 3 review (M4) will: confirm that the integrated ES (finalized Operational Prototype) satisfies the design and functional requirements and all computer programs and databases are fully developed; confirm that final T&E has been carried out satisfactorily and support products (e.g., documents and training courses) have been developed and reviewed; present and explain all major modifications to the Development Phase schedule, cost, and product; present the project schedule and cost estimation for the Deployment and Post-Deployment Phases; present an update on staffing, training, acquisition and fielding strategy, and ROI; present the detailed plan for carrying out the Deployment Phase; and present project support needs.

The result of the Milestone 3 review is: to continue the project by moving on to the Deployment Phase; or to iterate back to the Testbed Prototype or Operational Prototype to correct recognized inadequacies of the ES; or to terminate the project.

Overview of the Technical Process. The technical process begins by defining a detailed technical plan (T1) for developing the Operational Prototype. The number and characteristics of the prototype iterations will depend on the application interface and integration needs. For example, an ES that is loosely coupled to an AIS for read-only data is less of a danger in terms of potential disruption to the AIS than is an ES that will write data or be embedded in the AIS. The approach minimizes the danger to the operational on-line system and the schedule and cost risk by beginning with small steps of limited associated functionality that can be tested in isolation and eventually building up to the completed Operational Prototype. The technical process described below is an example of the steps that might be followed.

The detailed technical plan will describe each iteration or step as to: its objective in terms of the operational functionality it will be testing, its design to implement the functionality, and its estimated schedule and cost. The next step is a joint technical/management preparation of the T&E plan (J1) to specifically define what to test,

how to test, the test data needed and its sources, how test criteria are to be defined, who will conduct the tests, and who will evaluate the tests (e.g., specialists, experts, end-users).

After the T&E plan is drawn, the iterative development of the Operational Prototype (T2) begins. If the ES will be interfaced to more than one AIS and/or special I/O devices, the interfaces can be worked on concurrently. The steps in Operational Prototype development are:

- Step 1: establish low-level connections between the ES and special I/O devices and/or communications interfaces to AIS(s) and test them (e.g., by sending signals or sample communication packets).
- Step 2: establish high-level communication interfaces using test cases that were used to test the Testbed Prototype; verify that the answers are the same as for the Testbed Prototype final T&E, that timeliness of performance is adequate, and that security requirements have been met.
- Step 3: demonstrate high-level communication interfaces using "real" data to test the system for correctness of response and timeliness of performance.
- Step 4: demonstrate recoverability by causing failure in various ways, and test the adequacy of the system to meet recoverability and security requirements.
- Step 5: demonstrate robustness by running the Operational Prototype for many hours or days, and test for nondegrading performance in terms of timeliness and correctness of response.
- Step 6: run the finalized Operational Prototype using real data in parallel with the on-line system.

Each step or iteration will have a well-defined objective, design, T&E plan, schedule, cost, and IPR. Scheduled IPRs should be arranged at approximately 6-month intervals; several of the interfaces and steps could go through IPR at the same time. Any major problem arising during an iteration will trigger an unscheduled IPR. Each IPR will determine: if the project is proceeding well in meeting its plan and if it should continue in the current step or go on to the next step; if the Operational Prototype is complete; if the project should be terminated or if the plans should be changed (either to exclude the problem or to solve it by returning to an earlier step or to the Testbed Prototype). Specialists (e.g., in ADP, communications, devices, security) may participate in the development and in the T&E. Application experts and end-users will participate in steps 2-6. Participants in the IPRs will include the various specialists as well as experts, end-users, and user organization managers.

During the Operational Prototype development the various steps should be documented in the Progress Notebook. After step 6, that documentation should be used to develop the Users Manual (UM), the End User Manual (EM), the Maintenance Manual (MM), the Computer Operation Manual (OM), the Test Plan (PT), the Test Analysis Report (RT), the Implementation Procedures (IP), and product configuration control techniques (T3). The rest of the documentation—Functional Description (FD), System/Subsystem Specification (SS), Software Unit Specification (US), and Database Specification (DS)—should be updated and completed, and training courses may be developed for users, operations, and maintenance personnel.

The acquisition and fielding strategy (J2) for the Deployment Phase will be updated in a joint management and technical effort. Acquisition strategy, the Progress Notebook and product documentation, and training courses will be inputs to management preparation for the Milestone 3 review. The last technical task is to write an SOW (T4) for the Deployment Phase.

This chapter is divided into four sections: Section 8.1, this section, is an introduction; Section 8.2 presents an example risk containment table; Section 8.3 describes the management activities; and Section 8.4 describes the technical process.

8.2 RISK CONTAINMENT

Table 8.1 is an example prioritized list of major risk items that need to be identified and addressed during the Operational Prototype effort. Since applications vary, the project manager should develop his/her own list of the top ten risk items.

Table 8.1

AN EXAMPLE PRIORITIZED LIST OF OPERATIONAL PROTOTYPE RISK ITEMS

	Risk Item	Risk Management Techniques
1.	Personnel shortfalls	Staff this phase with top talent, especially specialists in ADP, security, and communications
2.	Unavailability of equipment	Ascertain availability of off-line AIS(s) and dedicated special I/O devices; if AIS(s) are also used for backup, factor into schedule and costs
3.	Unforeseen problems with integration	Perform integration in small steps with specific and thorough testing before proceeding ahead
4.	Performance problems	Selectively and extensively collect runtime data in order to analyze and find performance bottlenecks
5.	Security accreditation problems	Work closely with security specialists and people from accrediting agency to understand in detail what is required; have them review design and performance
6.	Failure to develop document products and training products on time	Make it clear to the staff which document and training products they are responsible for, and plan enough money and time for them to develop good products; assign a project librarian to be responsible for maintaining all documentation; PM should oversee schedule and review documents; ensure validation of documentation
7.	Gold plating	Avoid adding additional functionality; document and save for post-deployment upgrades
8.	Unrealistic schedule and expectations	Be realistic in plan and schedule; do careful design and decomposition into integration steps; recognize when cost and schedule are inadequate to meet needs; recognize and handle problems when they occur
9.	Changes in deployment needs not reflected in acquisition strategy	Work with user community and staff to quickly identify changes that affect deployment plans and modify acquisition strategy asap to avoid long delays in the deployment phase
10.	Effect of changes in requirements on validity of problem to user mission	Evaluate mission needs whenever considering major changes to problem, integration, security, or communications

8.3 MANAGEMENT ACTIVITIES

The management activities are organized into:

8.3.1 Initial Management Activities (M1)
8.3.2 Development of T&E Plan for Operational Prototype (J1)
8.3.3 Ongoing Management Activities (M2)
8.3.4 Acquisition Strategy Update (J2)
8.3.5 Preparing for the Milestone 3 Management Review (M3)
8.3.6 Holding the Milestone 3 Management Review (M4)

8.3.1 Initial Management Activities (M1)

The initial management activities are: (1) to develop a management plan; (2) to ensure that the project is adequately staffed with relevant specialists in ADP, security, communications, and maintenance as well as ES technology; (3) to provide staff, expert, and end-user training; (4) to ensure that necessary equipment is available and supported; (5) to carry out plans developed for QA, configuration management, deployment environment security, and maintenance; and (6) to define the documentation requirements for the Operational Prototype effort.

Management Plan. A management plan should be prepared that includes: technical plan milestones; schedule for acquiring staff; schedule for acquiring equipment; training schedule (for staff including ADP, security and communication specialists, experts, and end-users); documentation schedule; maintenance schedule; IPR schedule; and the schedule of monthly meetings to review progress and risk management. The plan should allow some slack in the schedule if an off-line AIS is being used that is also used as backup (i.e., because the AIS might not be available when needed).

Staffing, Training, and Equipment. The technical staff requirements were addressed in the SOW and should be reviewed with the KE or technical manager and modified if necessary. If the work is being done in-house and staff or specialists are unavailable, the PM may want to consider hiring qualified consultants.

Training is very important for all participants in the Operational Prototype effort. Various specialists, maintenance personnel, and site resident experts and end-users who have not received previous ES technology training will need to be trained. Training requirements are discussed in Section 5.3.3.8.

Equipment must be acquired in a timely fashion in order to carry out the development schedule and meet the technical plan milestones. Equipment includes not only the ES tool and workstations but also availability of off-line or backup AIS(s), special I/O devices, off-the-shelf or customized hardware/firmware/software interfaces, and test equipment.

Carrying out Plans. It is the PM's responsibility to see that plans developed for QA, configuration management, security, and maintenance are carried out.

- The PM should ensure that all technical staff are informed and/or trained as to QA guidelines and configuration management procedures.
- The PM should ensure that the project librarian has training and/or computer aids in order to best carry out the configuration management functions.
- If the deployed ES is to be a secure system, the PM must see that proper security measures are enforced during Operational Prototype development with regard to: physical environment (including workstations and disks); clearances for staff members, experts, users, maintenance personnel, and specialists; labeling of inputs and outputs; user/developer access and authorization procedures; etc. Part of the technical plan will be to demonstrate that the completed Operational Prototype is in compliance with DoD security policy and that the deployed system can be accredited or certified.

- The PM must plan for: ES tool/workstation installation and checkout; ongoing maintenance of all equipment; and either vendor, contractual, or in-house support for the ES tool and/or workstations.

Documentation Requirements. The PM should define the document products that are deliverables at the end of this phase and make them known to the technical staff. The types of documents the PM might require are shown in Figure 8.2 and were derived from Figure 3.2, "Automated information systems life-cycle documentation," taken from Draft DoD Standard 7935.1, 15 January 1988. It is recommended that the document requirements include: Functional Description (FD), either or both System/Subsystem Specification (SS) and Software Unit Specification (US), Database Specification (DS), Users Manual (UM), End User Manual (EM), Computer Operation Manual (OM), Maintenance Manual (MM), Test Plan (PT), Test Analysis Report (RT), Implementation Procedures (IP), and Deployment Phase SOW. The PM should specify sections in the Progress Notebook for all required documents as well as sections for information, issues, and problems not provided for in the above documents. Additional sections are needed for the: project management plan; project technical plan; "problem" descriptions and resolutions; and reports on IPRs and monthly management meetings.

The PM should designate a project librarian who will be responsible for maintaining the documentation.

8.3.2 Development of T&E Plan for Operational Prototype (J1)

Developing the T&E plan for the Operational Prototype is a joint technical and management activity. It is addressed in Section 8.4.2 as part of the technical process.

8.3.3 Ongoing Management Activities (M2)

Ongoing management activities include: carrying out the project plan and modifications made to it as a result of the prototyping effort; monitoring QA, configuration management, T&E, training, risks, security, and project support; facilitating exchanges across the developer, user, maintenance, ADP, and security and communications communities; and planning for deployment (e.g., site installations, acquisition of hardware and software, and training of users and maintenance personnel).

Managing the Project Plan. The objective of the Operational Prototype is to integrate the stand-alone Testbed Prototype into an example operational environment using the actual deployed ES tool, AIS(s), special I/O devices, and communications and telecommunications networks. The result will be a deployable ES that has demonstrated: integration capability, fulfillment of performance requirements, fulfillment of recovery and robustness requirements, compliance with DoD security policy, and ability to run parallel to the current on-line system.

Risks to the project plan include: off-line AIS(s), unforeseen interfacing problems, and gold plating. If the off-line AIS also serves as a backup system, the project schedule should allow for the fact that the AIS may not always be available to the project. Unforeseen dangers of integration are discussed at the beginning of this chapter. They have been partially addressed by attending to integration issues from the beginning. In this phase the risk can also be reduced by planning and carrying out integration in small, manageable steps that can be more easily tested and evaluated. Although most major issues should have been addressed by the earlier prototyping phases, working with end-users and real case data at a new site may expose functional problems and oversights. Gold plating (i.e, adding functionality to satisfy an additional need) is a possible risk in this phase and should be avoided whenever possible. However, any gold plating suggestions should be documented so they will be available in the Post-Deployment Phase to consider as part of the product upgrade.

As with the earlier prototypes, the project plan, schedule, and costs should be able to easily and inexpensively accommodate necessary technical changes. This requires careful

management oversight so that deviations from approved goals, schedule, and cost are quickly recognized and addressed without imposing excess management demands on the technical team. The two review techniques for achieving this are: informal monthly project reviews, and more formal, in-process reviews (IPRs).

The monthly reviews should concentrate on progress, risk management, and problems that occurred during the previous month. The review could be in the form of a short written report, a discussion between the PM and technical leader and perhaps a specialist or user organization representative, or a meeting of all or selected members of the project staff.

In-process reviews should be scheduled at approximately 6-month intervals. A scheduled IPR may address more than one interface or integration task if the tasks can be separated and concurrently developed. The scheduled IPR addresses whether the Operational Prototype is meeting its goals, the T&E results, problems to be resolved, risk management, revisions to plans, and the next step (e.g., go back to an earlier iteration, go on to the next step, conclude the project). Attendees may include representatives from all interested communities and should, as a minimum, include the project staff and the user organization manager.

An unscheduled IPR is held to address a major problem; attendance should be decided by the PM with recommendations from the KE or technical leader.

Activities that Need Monitoring. In addition to monthly reviews and IPRs, the PM should:

- Direct the KE or technical leader in monitoring QA.
- Monitor the configuration management activity and address any management problems that occur (e.g., technical staff evading the process).
- Monitor the T&E activity addressing any problems with: plan; acquisition of test cases; participation of the user community or specialists in ADP, security, and communications; and follow-up on technical response to results.
- Monitor all training activities by talking with people who have been trained and their managers to determine that the training is being done well and is sufficient. If not, the PM should look into making other arrangements.
- Monitor risks through monthly meetings. Be particularly aware of risks associated with changes to the deployment plans as these may require immediate attention to acquisition strategy.
- Monitor security policy and methods to ensure that they are being properly enforced in the development of the Operational Prototype. In addition, the PM should ensure that T&E confirms the Operational Prototype's compliance with DoD security regulations. Last, the PM should arrange for the completed Operational Prototype to undergo security accreditation or certification by the responsible DoD agency to ensure that the deployed system will not encounter unforeseen security problems.
- Monitor project support needs: maintenance of tools and equipment, office and laboratory space and usage, secretarial and documentation support, etc.

Facilitate Exchange Across Communities. The PM should, in general, facilitate information exchange across user, developer, maintenance, ADP, security, and telecommunications communities by keeping them informed and inviting them to relevant IPRs.

In particular, the PM should establish close ties with the user community at the Operational Prototype site. The PM has a responsibility to keep the site user-organization manager informed of project needs for expert and end-user participation, including training and T&E. The PM should frequently discuss user satisfaction with the user organization manager, experts, and users who are participating in the project. The project cannot succeed unless a good relationship is established between the user and developer organizations and unless the user organization accepts responsibility for its part of the project effort.

The PM should work with the maintenance community to develop an understanding of how maintenance will be carried out in the Deployment and Post-Deployment Phases. This

includes which levels of maintenance will be furnished on-site or through local vendors and which will be handled in a more centralized way. This is very dependent on the geographic dispersal of sites and users and the variability of their needs. For example, sites and/or users may differ enough in long-range plans to require more than one KBA. It is extremely important to understand these needs in order to develop the appropriate training courses and product documents for maintenance.

The Development Phase requires the participation of ADP and communications specialists in the development and T&E of integration and interfaces. If the ES is a secure system, security specialists are also needed. The PM will have been working with these communities during the earlier phases of development but will need more commitment from them during this phase.

The PM should continue to brief the project to higher management to attract champions and generate enthusiasm and support. A major PM responsibility is to ensure that user organizations targeted to use the ES are briefed and interviewed as to reactions, fears, suggestions, etc. These should be fed back to the developers and reflected in the user documentation and training course. If there are plans to deploy the ES to users at different sites, this is a good time to brief them on the project also. As mentioned in the Concept Phase, use of a videotape presentation to demonstrate the ES prototype is highly recommended.

Planning for Deployment. The PM should work with the user community to develop a detailed plan for deployment that will be supported by the acquisition and fielding strategy. This includes: the sites and number of users that will be using the ES; order of deployment; equipment and software acquisition needs; training courses for users, operators, and maintenance personnel; and requirements for KBAs.

8.3.4 Acquisition Strategy Update (J2)

Updating the acquisition strategy so that it effectively integrates the technical, business, and management elements of the project is a joint technical and management effort. It is discussed in Section 8.4.6 as part of the technical process.

8.3.5 Preparing for the Milestone 3 Management Review (M3)

The main purpose of the Milestone 3 review is to ascertain that the Operational Prototype satisfies all requirements for deployment and is valid in meeting the mission needs of the user organization.

Information to be presented at the review includes:

- Description of Operational Prototype compliance with the design and functional requirements.
- Project plan, schedule, and cost estimation for the Deployment and Post-Deployment Phases.
- An update on:
  - Staffing and training.
  - Acquisition and fielding strategy.
  - Return on investment.
- Detailed plan for carrying out the Deployment Phase.

Discussion of Operational Prototype Compliance with the Design and Functional Requirements. The PM should prepare a high-level walk through the design and show how it meets the functional requirements presented in the Functional Description (FD). This should concentrate on (1) the final integration and interfacing design and (2) ES changes since the Milestone 2 review. The PM should address compliance of the Operational Prototype with DoD security requirements and any remaining certification or accreditation problems that would prevent the ES from being deployed as a secure system. If such problems exist, possible solutions should be presented.

The PM should confirm that: all computer programs and databases have been fully developed; final T&E has been carried out; performance requirements have been met; and support products (i.e., documents and training courses) have been developed and reviewed. All major modifications to the Operational Prototype schedule, cost, and product occurring during the Development Phase should be presented and explained. If modifications have occurred, any effects they have on the validity of the ES to the user community's mission should be presented. A short videotape presentation of the final Operational Prototype may be an effective way to demonstrate the ES and its integration with AIS(s) and special I/O devices.

Discussion of Project Schedule and Cost Estimation for the Deployment and Post-Deployment Phases. Estimated schedules and costs for the Deployment and Post-Deployment Phases should be presented. This should include: the size and level of effort required for each phase; estimated amount of effort and cost for maintenance including the KBA(s) maintaining the ES and changing knowledge base; training costs; and the estimated cost of project management activities and reviews. Again, as in the previous cost estimates, the costs of other similar efforts should be compared with the new cost estimate to ensure that it is not off by an order of magnitude or more.

Update of Staffing Needs. Experience in developing the Operational Prototype and in writing the Implementation Procedures (IP) manual should result in a more accurate assessment of the type and number of staff required to deploy the ES and participate in post-deployment activities. Likewise, a more accurate assessment can be made of the number and tasks to be performed by various maintenance and support personnel.

Update of Training Needs. Training needs and plans should be well defined as a result of experience in: developing the Operational Prototype; producing the End User Manual (EM), Computer Operator Manual (OM), Maintenance Manual (MM), and Configuration Control Manual; and training courses.

Update of Acquisition and Fielding Strategy. The updated acquisition and fielding strategy should be presented. The new strategy should accommodate any changes in the plan to acquire hardware, firmware, and software for deployment. This will include any special software/hardware needed to interface to AIS(s) or communications networks as well as changes in special I/O devices. Advantages to buying in large batches should be addressed as well as any problems such as availability of the equipment, vendor support, and maintenance. The acquisition and fielding strategy must be consistent with the deployment plan.

Update of the Cost Benefit/ROI. The PM should refine the cost benefit/ROI presented at the Milestone 3 review based on experience during the Development Phase. Examples of changes that could have affected the ROI are: unforeseen problems in integration may have caused higher cost; overlooked functional needs may have required changes to the testbed ES at additional cost; overlooked functional needs may have required a narrowing of the problem scope resulting in less benefit to the user mission; and costs may have been higher than anticipated for specialists, equipment, training, security, maintenance, or communications.

Discussion of Detailed Plan for Carrying out the Deployment Phase. Describe the detailed plan that will be included in the SOW for the Deployment Phase. It should include: number of sites and users; when deployment is scheduled to start and to complete at each site; training schedules for users, operators, and maintenance personnel at each site; and how the implementation procedures will be carried out. Deployment will include complete testing of the system and, for secure systems, accreditation or certification by the relevant DoD agency. The plan should include identification of the person at each site responsible for acceptance of the deployed system and all product materials.

8.3.6 Holding the Milestone 3 Management Review (M4)

The last management activity of the Operational Prototype effort is to present the project at the Milestone 3 Management Review. Managers at all levels of the participating organizations (especially those representing user sites) should be encouraged to attend, as

well as expert and end-user representatives, representatives from the operations and maintenance organizations, and specialists in ADP, security, or communications who participated in the Operational Prototype effort. As mentioned earlier, a videotape may be an effective way of describing the finalized Operational Prototype.

The result of the Milestone 3 review will be: to deploy the Operational Prototype; or terminate the project; or to iterate back to the Testbed Prototype or an earlier Operational Prototype iteration because of inadequacies in meeting the functional requirements.

8.4 TECHNICAL PROCESS

The technical process of the Operational Prototype can be organized into the following tasks:

8.4.1 Technical Plan for Operational Prototype Development (T1)
8.4.2 Development of T&E Plan for Operational Prototype (J1)
8.4.3 Development of Operational Prototype (T2)
8.4.4 Scheduled and Unscheduled IPRs (M5)
8.4.5 Development of Final ES Products (T3)
8.4.6 Acquisition Strategy Update (J2)
8.4.7 The SOW for Deployment Phase (T4)

8.4.1 Technical Plan for Operational Prototype Development (T1)

The technical process begins by defining a detailed technical plan for developing the Operational Prototype. The number and characteristics of the prototype iterations or steps will be dependent on the interface and integration needs of the application. If the application requires interfacing to or integrating with more than one device or AIS, each effort can be performed independently and concurrently and then brought together into the Operational Prototype at an appropriate time.

The technical plan should minimize the risk to current on-line operations and to schedule and cost by beginning with small steps of limited functionality that can be tested in isolation, and eventually building up to the completed Operational Prototype. During this process stubbed-in interfaces in various Testbed Prototype modules may be reimplemented, and software and/or hardware interface modules may be acquired or developed in accord with the concept prototype design. Section 8.4.3 describes six incremental steps in carrying out an example integration interface.

The detailed technical plan will describe each iteration or step as to: its objective in terms of the operational functionality it will be developing, its design to implement the functionality, its estimated schedule and cost, and time of the scheduled IPR. Test and evaluation should be carried out after each iteration or step. For review purposes, IPRs should be scheduled at approximately 6-month intervals, though of course unscheduled IPRs will be held whenever a major problem arises.

The cost and schedule to carry out the technical plan should be realistic. If the cost and/or schedule are fixed and inadequate to accomplish the work, then the technical leader should indicate so to the PM and the issue should be resolved before any implementation takes place. If the application has to be reduced in scope in order to meet the cost and schedule constraints, then the newly scoped application should be re-evaluated to be sure it still meets mission needs. If it does not, the project may be terminated.

8.4.2 Development of T&E Plan for Operational Prototype (J1)

After the technical plan is completed, the T&E team with the help of the PM and specialists should develop detailed T&E plans. Specialists in ADP, special I/O devices, communications, and security should be made available and become a part of the T&E team during this phase. The T&E plan needs to address: *what* is to be tested; *how* the prototype

ES is to be tested; a definition of the test data and their sources; *how* the criteria for the tests are to be established; *who* will conduct the tests; and *who* will participate in the evaluation.

What Is To Be Tested. The T&E team should define precisely what is to be tested in each step or iteration. For example, an AIS interface may be tested at a low level to be certain, (1) that the prototype network controller is sending/receiving correct packets to/from the network connected to the AIS; (2) that it is doing so within timing requirements; and (3) that it can handle errors correctly. At a higher level, the interface may be tested to ensure (1) that it is correctly sending/receiving data used in T&E of the Testbed Prototype; (2) that the data received are being correctly used by the ES and that data sent from ES to AIS are correct; and (3) that errors generated by the test program are handled correctly. Next, testing may be done using captured "real" data presented to the ES in real time. Tests are needed to determine the robustness, recoverability, security, and performance of the Operational Prototype.

How the Prototype ES Is To Be Tested. The T&E plan must specify how the ES will be tested. Special test equipment or software routines may be needed to capture the input and output at each end, to capture performance and timing data, and to cause errors in order to test error handling and recovery. The testing equipment may be designed, implemented, and run by specialists who are part of the T&E team, or this may done by the technical staff and reviewed by the T&E specialists to be sure that the equipment tests what it is intended to test and gathers the data needed for evaluation.

Definition of the Test Data and Their Sources. The T&E team must determine what test data are needed and where they can be acquired. In some cases, for example in testing low-level interfaces, it may be necessary to program sequences of messages to test that the interface is generating the right network packet descriptors, that various-length messages (especially at the limits) are handled properly, and that errors are detected and handled appropriately. The programs that generate the data may be written, debugged, and run by a communications specialist. The source of the data may be a manual that describes the network protocol.

For higher-level tests the T&E team may conduct tests using the same database of test cases that was used for T&E on the Testbed Prototype, but the criteria may have been extended to include timing requirements. When "real" data are used, the T&E team must describe how they will be acquired and used, how the results will be collected and analyzed, and how the "real" data cases will become additions to the T&E test database.

Establishment of Criteria for Test Data. The criteria for the test results may be defined in many different ways. For low-level interfaces, they may be defined from the operational manual or protocol document. For higher-level tests, the functional criteria will be determined in the same way they were for the Demonstration and Testbed Prototype developments—by qualified experts.

Who Will Conduct the Tests. Test and evaluation team responsibility for conducting particular tests will depend on what is being tested and the specialities of the T&E team members. For example, tests of low-level connectivity should be conducted by T&E staff with ADP, communications, and security expertise, whereas higher-level tests can be conducted by the T&E staff who tested the Testbed Prototype.

Who Will Participate in the Evaluation of the Test Results. The people who will participate in the test evaluation are determined by the type of test. For low-level tests, T&E specialists in ADP, communications, and security should evaluate the test collection techniques and results—if the testing software/hardware was developed by the technical staff. If the testing software/hardware was developed by T&E specialists, then other specialists with appropriate experience should evaluate the software/hardware and test results. For a secure ES, security accreditation of the completed Operational Prototype will have to be performed by the responsible DoD agency.

8.4.3 Development of Operational Prototype (T2)

After the T&E plan is developed, the iterative development of the Operational Prototype begins. As was pointed out earlier, if more that one interface is needed, each interface development effort could begin separately and, when appropriate, be brought together into a single Operational Prototype.

The steps for accomplishing each integration or interface will be highly dependent on the application. For illustrative purposes we describe below six steps in the integration of an ES with an AIS. In the example, the ES needs to receive data from the AIS, process them, and send new data back to the AIS. The AIS and ES are continuously running systems (i.e., 24 hours a day, 7 days a week) and need to process various categories of transactions within required time constraints. A fairly stable level of ES performance is required and needs to be maintained through cache management and garbage collection of data received from the AIS; through processing within time constraints; and through handling messages within time constraints.

The six steps (iterations) are:

- Step 1: establish low-level communication interface
- Step 2: establish higher-level communication interface using test case data
- Step 3: establish higher-level communication interface using "real" data
- Step 4: demonstrate recoverability
- Step 5: demonstrate robustness
- Step 6: run the finalized Operational Prototype with real data in parallel with the on-line system

These steps assume availability of an off-line AIS. This could be a backup system used in case of failure or used for testing new releases. An off-line system must be available unless the integration is loose (the Operational Prototype only reads from the AIS), and unless the risk of causing a failure of the AIS and interrupting its service is acceptable. An off-line system that is also used for backup will not be available for dedicated project use, and interruptions to the Operational Prototype schedule should be taken into account during planning. Note that many of the steps involve the collection and analysis of runtime information in order to demonstrate or validate that the system is operating correctly or within timing constraints. The collection of the runtime information, in many cases, will exert an effect on the system performance and needs to be factored into the analysis.

If problems are found during any of the steps, problem reports should be documented in the Progress Report. Any minor problems that can be corrected within cost and schedule should be corrected; any major problem will require an unscheduled IPR.

Step 1: Establish Low-Level Communication Interface. In this step the low-level communication interface between the Operational Prototype and the AIS is implemented (or the low-level interface between a special I/O device and the Operational Prototype could be implemented) using the off-line AIS. The interface should demonstrate that correct data packets can be sent from the AIS and received by the Operational Prototype, performance and error handling is acceptable, different size data requests can be handled correctly (e.g., test from the extremes of 1 byte to megabytes), different data flow rates can be handled correctly (test for errors that would flood the network), and performance is within acceptable bounds (no unexplained delays). The same characteristics should be demonstrated for data packets being sent from the Operational Prototype to the AIS. Finally, robustness should be tested (1) by exchanging packets for long time periods and (2) by pulling the plug on the network, AIS, and Operational Prototype and then testing recovery procedures.

Test and evaluation should be carried out at the completion of this step. Any minor problems that can be corrected within cost and schedule should be corrected. Major problems (e.g., need to rewrite the interface) will require an unscheduled IPR. Successful conclusion of this step demonstrates that low-level communication between the Operational Prototype and AIS works.

Step 2: Establish High-Level Communication Interface Using Test Case Data. After step 1 has passed T&E, then the higher-level functions of the Operational Prototype can be demonstrated using the test cases that were used for T&E of the Testbed Prototype. The test case data should be put on the off-line AIS in order to demonstrate that they can be correctly read from the AIS to the Operational Prototype. They should be processed on the Operational Prototype and the answers verified. The updates or new data output from the processing should be sent to the AIS and verified there. If the criteria include timing requirements, then timing information should be collected on the AIS, network, and Operational Prototype and analyzed. The security aspects of the system need to be demonstrated regarding: physical security; user authorization; marking of classified output screens and printed information; and data, computer, and system security. The goal is to pinpoint and address areas where (1) performance bottlenecks occur due to either error or poor design, or (2) security requirements have not been met.

Test and evaluation should be carried out at the completion of this step. Any minor problems that can be rectified within cost and schedule should be corrected. Major problems (e.g., requiring a design change to meet a timing constraint) will require an unscheduled IPR. Successful conclusion of this step demonstrates that (1) the Operational Prototype handles the interface and the test cases correctly, within timing constraints, and (2) the security requirements have been met.

Step 3: Establish High-Level Communication Interface Using "Real" Data. Step 3 uses the communication interface demonstrated in step 2 to test how well the Operational Prototype handles "real" data. Captured "real" data from the on-line system is entered on the off-line AIS to be read by the Operational Prototype. The "real" data will need to be evaluated by experts as part of T&E, and criteria established in order to determine if the data are being properly processed by the Operational Prototype. Examples of serious problems would be incorrect handling of "real" data or discovery that a category of data had been overlooked during problem definition.

Test and evaluation should be carried out at the completion of this step. As part of the T&E, local end-users who have processed the "real" data cases in the current on-line system should process the same "real" data cases using the Operational Prototype. Their evaluations will help determine how well the Operational Prototype, especially the user interface, meets their needs. It is important to resist gold plating at this point. End-user evaluations should focus on correcting problems with the Operational Prototype, not adding functionality. Suggestions for additional functionality should be written up in the Progress Notebook where they can be considered in post-deployment upgrades. Any minor problems that can be rectified within cost and schedule should be corrected. Major problems (e.g., correcting the processing of a particular case category) will require an unscheduled IPR. Successful conclusion of this step demonstrates that the Operational Prototype handles "real" data cases correctly. The real data cases should be added to the T&E test database.

Step 4: Demonstrate Recoverability. Now that the Operational Prototype in off-line mode has demonstrated that it is capable of handling "real" data cases, the next step is to try to break it in as many ways as possible to demonstrate its recoverability characteristics. These tests should include: power outage; attempting to violate security policy and mechanisms; pulling plugs on the Operational Prototype's disks, workstation, network connection, AIS; causing probable errors to occur with the data themselves (e.g., wrong data values, missing data, duplicate data); and detecting failures in sending and re-sending messages (e.g., detecting too many messages or too many re-sent messages, detecting when the network is not responding). Discovery of a major recovery problem will require an IPR.

Test and evaluation should be carried out at the completion of this step. As part of the T&E, specialists should examine the design of the recovery mechanisms as well as the results of testing. Any minor problems that can be rectified within cost and schedule should be corrected. Major problems (e.g., redesign and re-implementation of a recovery mechanism) will require an unscheduled IPR. Successful conclusion of this step demonstrates that the Operational Prototype can recover from the failures it was tested for.

Step 5: Demonstrate Robustness. The next step after demonstrating recoverability is to demonstrate robustness. The assumptions state that the Operational Prototype and AIS will be running continuously. This step must demonstrate that the Operational Prototype and AIS can run continuously for a test period of, for example, 7 days without degradation in performance (i.e., timeliness and correctness of response) and in cases of failure can recover as previously demonstrated. This will require having end-users operate the system.

Runtime information needs to be collected because what needs to be demonstrated is the system's ability (1) to manage its data cache using garbage collecting techniques and (2) to keep up with the demand on the system (maybe by selectively processing only certain data cases or users when the load gets too heavy). The process of collecting the runtime information can, in turn, affect the runtime performance and thus needs to be taken into consideration when analyzing performance. It is very important to make sure that the data cases are extremely varied. If possible, "real" data cases should be captured and used. If the detailed analysis of 7 days' worth of such data is too costly, statistical methods can be used to select meaningful samples to analyze in detail. Graphs of performance over the runtime period can help identify possible trouble periods that will require a more detailed examination.

Test and evaluation should be carried out at the completion of this step. As part of the T&E, specialists should examine the runtime collection routines to ensure that the proper information is being captured. Any minor problems that can be rectified within cost and schedule should be corrected. Major problems (e.g., redesign and re-implementation of a cache management mechanism) will require an unscheduled IPR. Successful conclusion of this step demonstrates that the Operational Prototype and off-line AIS can successfully sustain operation over the stated test period, thus demonstrating system robustness.

Step 6: Run the Finalized Operational Prototype with Real Data in Parallel with the On-line System. This is the last step in the development of the Operational Prototype. The goal of this step is to run the Operational Prototype in parallel with the existing on-line system. To do this in a data-driven Operational Prototype, one needs to capture and port the data from the on-line system to the prototype system in real time or close to real time. If the system is user-driven, true parallelism may be impossible or difficult to achieve. The best that can be done might be to give the prototype users the same tasks that on-line system users were given.

This step should be conducted in two parts. The first part should have runtime performance collection turned on in order to demonstrate that the Operational Prototype can meet or be close to meeting performance criteria. The second part should be run with runtime collection turned off or removed so that the Operational Prototype is running as it will when it goes on-line.

Test and evaluation should be carried out at the completion of this step. In particular the end-users should do a final evaluation of the user interface, performance, and support features. At this time the DoD agency responsible for accrediting or certifying secure systems should be requested to accredit or certify the Operational Prototype. Any minor problems that can be rectified within cost and schedule should be corrected. Major problems (e.g., change to the security mechanism, added functionality, change to user interface) will require an unscheduled IPR. Successful conclusion of this step demonstrates that the Operational Prototype development is completed—the Operational Prototype can operate in on-line mode.

General. Each step or iteration will have a well-defined objective, design, T&E plan, schedule, cost, and scheduled IPR. Several of the interfaces and steps may go through IPR at the same time. The scheduled IPRs should be arranged at approximately 6-month intervals. Any major problem arising during an iteration will trigger an unscheduled IPR. Each step should be well documented in the Progress Notebook because the documentation will be input to the final products.

8.4.4 Scheduled and Unscheduled IPRs (M5)

During this phase of development, for an application requiring interfacing to or integrating with more than one device or AIS, each effort can be performed as an independent concurrent task and then brought together into the Operational Prototype at an appropriate time. Scheduled IPRs will be held at approximately 6-month intervals to review the status of one or more of these Operational Prototype integration tasks.

An unscheduled IPR is called when a major problem is identified that may cause considerable change to the plans and schedule. Unscheduled IPRs may be called any time during development of the Operational Prototype or as a result of T&E performed at the end of each integration step.

The people who should attend the IPR include: all project staff and managers; the user organization managers; maintenance personnel; experts and end-users who have participated in the tasks being reviewed; and specialists in ADP, communications, and security who have participated in the tasks being reviewed.

Unscheduled IPR. The unscheduled IPR should address the current status of the project in meeting its goals and objectives, the problem that prompted the IPR, alternatives for its solution, and recommendations from the technical team. The PM (with recommendations from the KE) should decide whom to invite to the unscheduled IPR.

What to do about the problem will depend on a number of factors:

- How well the project is going.
- How important the problem solution is with respect to the worth of the application to the user community and its mission if the problem is not solved. If it is worth solving then:
  - Availability of additional funding,
  - Acceptability of lengthening the schedule and/or finding additional staff.
- The amount of risk involved in the solution with respect to straining technology (ES, ADP, security, telecommunications, etc.).
- The degree of "gold plating" (capabilities outside the defined problem scope) required by the solution of the problem.

Alternative outcomes of the IPR are:

- To rescope the application to eliminate the need to solve the problem, to change relevant plans, and to continue operational prototyping.
- To obtain additional funding and a schedule extension and/or additional staff to solve the problem, to change relevant plans, and to continue prototyping effort (may require return to an earlier prototyping phase).
- To declare the current Operational Prototype iteration to be the end result and terminate the project.

The IPR proceedings and outcome should be reported in the Progress Notebook.

Scheduled IPR. The scheduled IPR should include:

- A review of the project's progress to date and how well it is meeting its objectives (this should include reference to all major and minor problems addressed since the previous IPR);
- A review of the T&E results, including an analysis of the T&E results by relevant ADP, security, or communications specialists or domain experts;
- A review of any problem uncovered by the IPR to be handled as described under "Unscheduled IPR."

Possible outcomes of the scheduled IPR are to accept the Operational Prototype as meeting its goals and:

- If this is not the final step, then proceed to the next step.

- If this is the last step and final T&E has been done then prepare for the Milestone 3 review.
- Terminate the project.

The IPR proceedings and outcome should be reported in the Progress Notebook.

8.4.5 Development of Final ES Products (T3)

Each step in the development of the Operational Prototype has been documented in the Progress Notebook. After step 6, that documentation should be used to develop the:

- Users Manual (UM)
- End User Manual (EM)
- Maintenance Manual (MM)
- Computer Operation Manual (OM)
- Test Plan (PT)
- Test Analysis Report (RT)
- Implementation Procedures (IP)
- Product Configuration Control Techniques

Other documents that should have been maintained up to this point and need to be completed as deliverables are:

- Functional Description (FD)
- System/Subsystem Specification (SS) and/or Software Unit Specification (US)
- Database Specification (DS)

In addition, training courses and related material should be developed for end-users and operations and maintenance personnel.

Short descriptions of the documents are given below.

Users Manual (UM). The UM presents general information about the ES and is directed toward the user organization's general management and staff personnel who have no need for detailed technical information concerning ES implementation or operation. It corresponds roughly to sections 1 and 2 of the UM described in DoD Standard 7935.1.

End User Manual (EM). The end-user manual describes how to use the ES in detail from an end-user perspective, including description of the workstation and how it operates. It should include a step-by-step tutorial that, if possible, can be run by the end-user on the ES workstation in demonstration or educational mode. The best way to organize the EM depends on the application and the end-users' view of the application. If the end-users conceptualize the system better from an application category view than from a functional view, then the category organization should take precedence over the functional view, or the EM might cover each in a different section. Sections that should be included are: help, browsing through the knowledge base, error messages, automatic recovery techniques, and the means by which the user can comment on or express his/her dissatisfaction with the system (e.g., problem with knowledge or reasoning, awkward interface, poor browsing, etc.). Error handling must include all error messages the end-user could receive from the ES tool, operating system, and network as well as from the ES. The end-user should be told at least where the error message came from and when it is necessary to tell a support person about the error.

Maintenance Manual (MM). The MM presents general information and specific technical information about the ES program for the Knowledge Base Administrator, the ES tool smith, and the ADP and/or network personnel who will maintain the interfaces.

Computer Operation Manual (OM). The OM contains precise and detailed information on the control requirements and operating procedures necessary to successfully initiate, run, and terminate the ES. The OM is directed toward supervisory and operator personnel who are responsible for the efficient performance of the ES. All recovery and backup procedures should be specified. Since the ES can fail at several levels (i.e., ES, ES tool, workstation, and

AIS and I/O device interfaces), the OM should describe how to ascertain, through the use of diagnostics and tests, the level at which a problem or failure is occurring and therefore what special help is needed. In general, the OM should be written in a step-by-step fashion as opposed to an expository style in order to clarify and emphasize the procedures associated with the ES operation.

Test Plan (PT). The PT is a tool for directing the ES testing and contains the orderly schedule of events and list of materials necessary to effect a comprehensive test of the ES. The set of test cases and criteria used in T&E shall be included in the list of materials. Those parts of the PT directed toward the user staff personnel shall be presented in non-technical language, and those parts directed toward the operations personnel shall be presented in suitable terminology.

Test Analysis Report (RT). The RT describes the status of the ES after testing and provides a presentation of deficiencies for review by staff and management personnel. All additional functionality requests from "problem" reports and as the result of T&Es, that were documented in the Progress Notebook, should be documented here. The RT should be prepared in non-technical language. The rationale for additional functionality should be clearly stated.

Implementation Procedures (IP). The IP is a tool for directing ES installation at locations other than the test site after ES testing has been completed. It may also be used to direct the implementation of major modifications or enhancements of an ES that has already been installed.

Product Configuration Control Techniques. This document describes the installed configuration control system that should be used to maintain the ES. It will be used by the Knowledge Base Administrator and by the ADP and communications personnel who maintain the interfaces. All configuration management information accumulated since the start of the Testbed Prototype development should be included in this document.

Functional Description (FD). The FD reflects the definition of the system requirements and provides the users with a clear statement of the system's operational capability.

System/Subsystem Specification (SS). The SS may be used to describe small systems or large system developments that can be broken down readily into subsystems, each with a separate area of responsibility. This document should be as detailed as possible concerning the environment and design considerations and should present more detailed information than the FD. In particular, this document should define the system/subsystem interfaces.

Software Unit Specification (US). The US is a technical document that describes in detail the program or program unit design and implementation.

Database Specification (DS). The DS is a technical document that describes in detail the data and knowledge used by the system, their source, acquisition techniques, system representation, where and how they are used, error handling techniques, removal/archiving methods, assumptions about the data, and implications of the data structure. All the information collected about source data should be documented in the DS.

8.4.6 Acquisition Strategy Update (J2)

The acquisition and fielding strategy for the Deployment Phase will be updated in a joint management and technical effort.

The PM may need to alter the earlier acquisition and fielding strategy as a result of changes in the deployment plans. Types of changes include: number of deployment sites, number of users, equipment base (e.g., workstations), deployment schedule, availability of maintenance personnel including KBAs, materials required for the user courses, and scheduling of training courses.

The PM may also want to change the acquisition strategy for carrying out the Deployment Phase based on experience acquired during the Development Phase. If the Operational Prototype effort was contracted out, the PM may or may not want the same contractor to continue on to the Deployment Phase or there may be reasons for preferring to do deployment in-house.

8.4.7 The SOW for Deployment Phase (T4)

After the Milestone 3 Management Review has been held and the Deployment Phase has been given the go-ahead, an SOW should be written whether deployment will be supported as an in-house effort or contracted out.

The SOW should include:

- Description of the ES, ES tool, workstation(s), the AISs and special I/O devices with which the ES interfaces, the interfaces, data and knowledge, and security and communication requirements. These can be taken from the relevant product documents.
- The deployment plan indicating the deployment sites and when deployment is scheduled to start and end at each site. This will include the number of users at each site, and plans for training users, operators, and maintenance personnel (including KBAs if appropriate). This information can be taken from the acquisition and fielding plan and from the training courses and their materials.
- A high-level description of the implementation procedures to be followed taken from the IP document, and the tests to be performed and test cases to be used taken from the PT and UM documents. This should include information on the availability of government specialists and maintenance people and of experts and end-users.
- A description of the product configuration control techniques in place to be used to maintain the ES.

If the Deployment Phase is to be competitively bid, then qualification requirements of the contractor should be stated in the RFP. These include:

- Expert-system development experience by project, indicating in more detail those projects using the specified ES tool and requiring similar integration.
  - Description of project, including current state of effort and integration ease/difficulty.
  - Project size in terms of ES tool used.
  - Project size in terms of staff, person-years, and cost.
  - Client contact person.
- System integration experience with respect to the ES tool and specific AIS(s), I/O devices, security, and telecommunications needs of the ES to be deployed, including:
  - Description of project, including current state of effort.
  - Project size in terms of person-years and cost.
  - Identification of key personnel available to work on the project.
  - Client contact person.
- Relevant domain experience including:
  - Projects performed.
  - Experience of identified personnel.
- Experience levels with the ES tool being used.
  - Indication of whether the tool is in-house and available for use.
  - Size and experience of the ES staff.
  - Identification of key personnel available to work on the project.

9. DEPLOYMENT PHASE[7]

9.1 INTRODUCTION

Figure 9.1 shows the place of the Deployment Phase in the expert system development process. The major technical and management activities for the Deployment Phase are shown in Figure 9.2. The gray area in that figure indicates the integration steps of this phase. During the Deployment Phase: the Operational Prototype is installed and tested at all user sites; training is conducted for all users, operators, and maintenance personnel (including KBAs); all products are updated; and system management transfers from the project manager (PM) to the organization responsible for its support through the remainder of its life cycle.

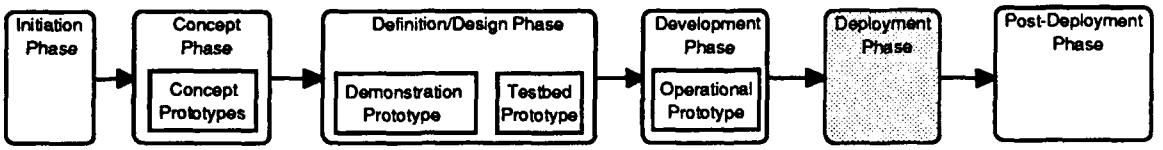


Figure 9.1—Expert system development process, Deployment Phase

The final Operational Prototype is the baseline system for the Deployment Phase. Any changes to the baseline should be made through a formal process for handling engineering change requests (ECRs) and engineering change orders (ECOs). This process should be used by the development project during deployment and may then be used by the support organization in the Post-Deployment Phase. To maintain the ES, it may be necessary to iterate back to prototype phases before the baseline. For example, small bugs can be eliminated or modifications properly made to the Operational Prototype; but larger changes in the knowledge base or knowledge base processes requiring ES tool development support may have to be (1) implemented and tested as changes to the stand-alone Testbed Prototype, (2) ported to the baseline Operational Prototype for further testing, and (3) finally released and re-installed at deployment sites. In the Post-Deployment Phase, major upgrades may require, for example, building Concept Prototypes or using the existing Testbed Prototype as a base for developing a new Demonstration Prototype and then reiterating back up through the various phases of development. For a mature ES, different phases of upgrade could conceivably be under way concurrently, each at a different phase stage.

For some ES tools the switch from development version to runtime version may be as simple as changing parameter settings. If so, the Operational Prototype can support knowledge base changes by using the development version of the Operational Prototype as an aid in debugging the ES and in modifying the interfaces and the runtime version for testing the performance characteristics of the system. Such a system makes life-cycle maintenance much simpler since it does not require (1) the use of the Testbed Prototype to maintain knowledge-related changes or (2) porting the updated testbed to the Operational Prototype.

The kinds of problems that may be encountered during deployment include: site-specific ES needs (e.g., knowledge base or hardware) that may require maintaining multiple

[7]See Appendix A for definitions of acronyms and terms used in this chapter.

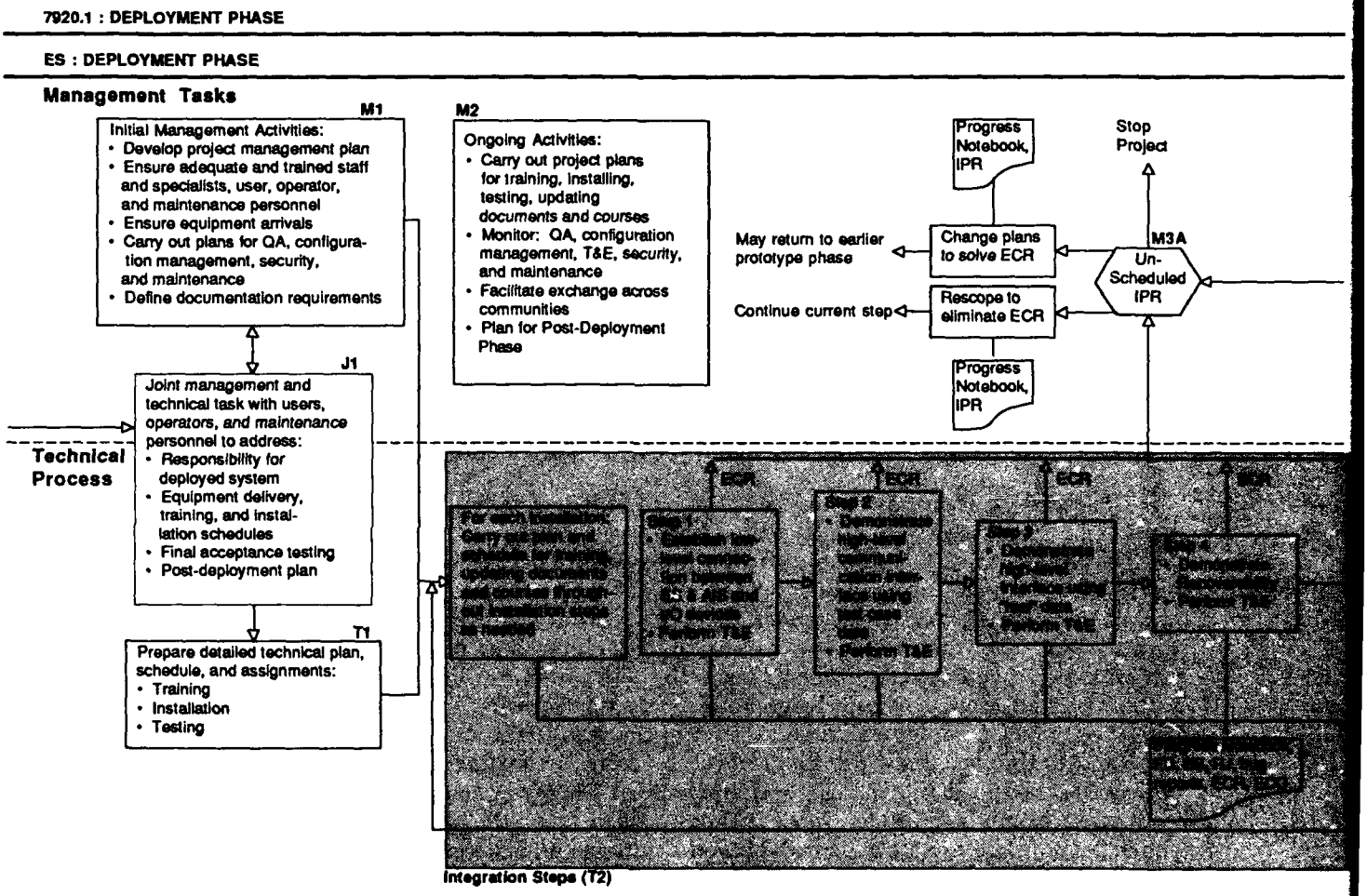
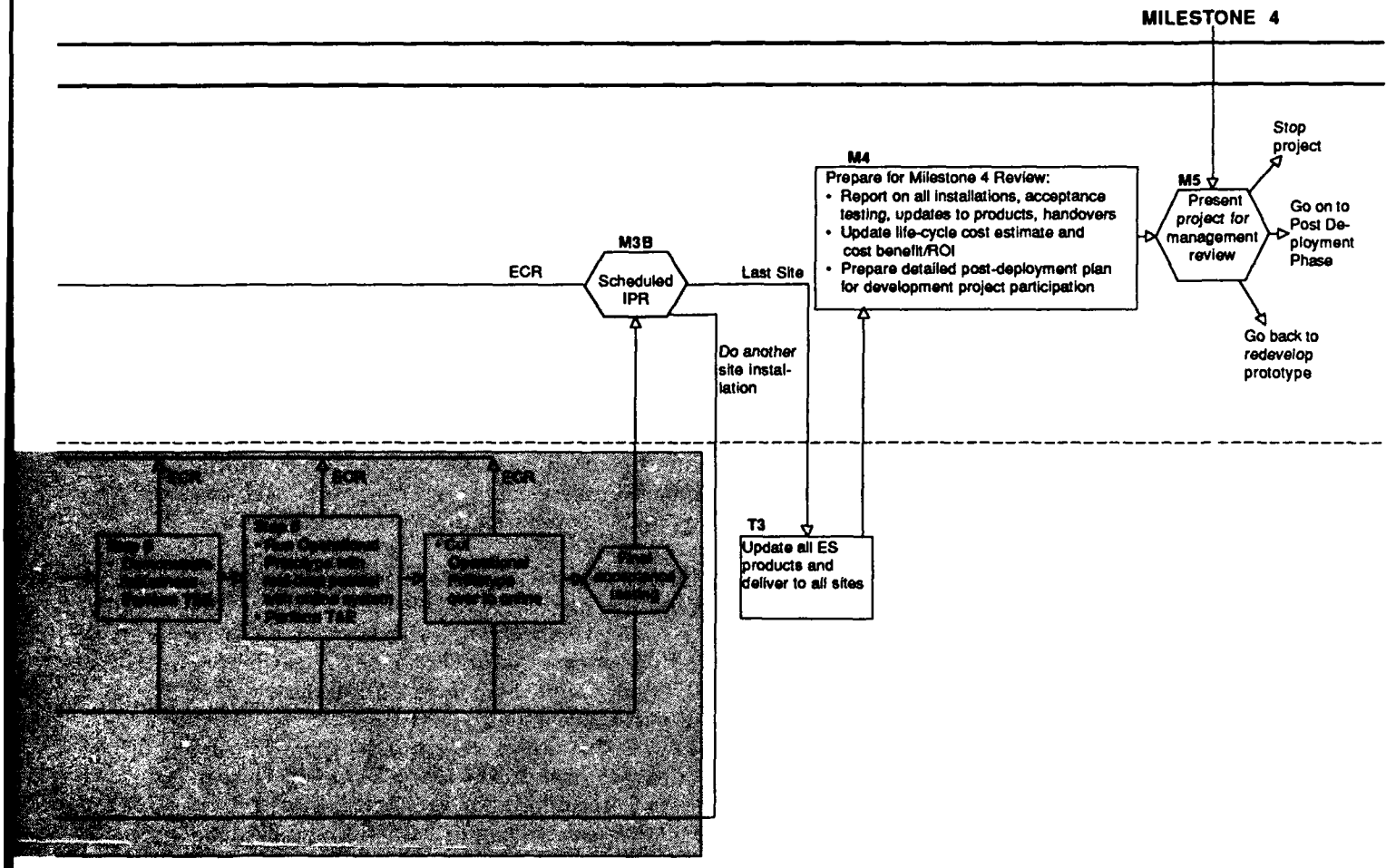


Figure 9.2—Deployment Phase


MILESTONE 4
M4
Prepare for Milestone 4 Review:
• Report on all installations, acceptance testing, updates to products, handovers
• Update life-cycle cost estimate and cost benefit/ROI
• Prepare detailed post-deployment plan for development project participation
M5
Present project for management review
Stop project
Go on to Post De-ployment Phase
Go back to redevelop prototype
M3B
Scheduled IPR
ECR
Last Site
Do another site instal-lation
T3
Update all ES products and deliver to all sites
ECR
ECR
ECR

versions of the ES; oversights due to (1) lack of complete problem definition and (2) poor choice of T&E test cases; and errors. Part of the planned deployment may be to accommodate site-specific needs by developing and maintaining multiple versions of the ES.

During this phase the Operational Prototype will be installed at each site, pass off-line acceptance testing, be brought on-line, pass on-line acceptance testing, and be handed over to the support organization responsible for the deployed ES. Essentially the same T&E steps are followed as for the Operational Prototype; an additional seventh step turns the ES on-line.

Overview of Management Activities. The first activity in this phase is to make detailed and explicit deployment plans (J1). This requires the joint participation of the management and technical development teams as well as participation of representatives from the user, operations and maintenance organizations.

This task specifically addresses: the person at each site to whom the deployed system will be handed; a detailed, phased training and installation plan for deployment at each site; description of the process and terms of the final acceptance testing (what it will be, who will run it, and how problems will be addressed); and a detailed post-deployment plan for development team responsibilities and activities.

Following this task, the PM should (M1): develop a project management plan; ensure that the project has adequate and trained staff personnel (including specialists, user participants, and maintenance and operating personnel); ensure that the equipment has been received or is in process and on schedule; carry out plans developed for QA, configuration management, security, and maintenance; and define the documentation requirements for the deployment effort.

Ongoing management activities (M2) include: carrying out the project plan for training, installation, testing, and updating documents and courses; monitoring QA, configuration management, T&E, security, and maintenance; facilitating exchanges across the developer, user, operations, maintenance, ADP, security, and communications communities; and planning for post deployment.

The PM will conduct in-process reviews (IPRs) at the end of each site installation (M3B) or by demand when a major unanticipated problem arises (M3A). Preparation for the Milestone 4 review (M4) includes: preparing a report on ES installations at all sites (particularly addressing training, acceptance testing, updates to products, and transference to the party(s) responsible for operating the ES); presenting the updated ROI and estimated life-cycle costs; and presenting a detailed plan for post-deployment.

The result of the Milestone 4 review (M5) is: to formally accept the deployed ES and continue with the Post-Deployment Phase; or to iterate back to the Operational or Testbed Prototypes to correct recognized ES inadequacies; or to terminate the project.

Overview of the Technical Process. The technical process begins by defining a detailed technical plan, schedule, and assignments for deploying the ES at required sites (T1). The Implementation Procedures (IP) developed during the Operational Prototype effort will be used as the installation guide and manual. It will include the T&E procedures to be carried out at each step. The example given below is a seven-step process (T2); the first six steps are those described in implementing and testing the Operational Prototype. The seventh step actually turns the system on and performs final acceptance testing.

Step 1: install equipment (e.g., special I/O devices and interfaces to AIS(s)) and establish and test low-level connections.

Step 2: demonstrate and test high-level communication interfaces using test cases.

Step 3: demonstrate high-level communication interfaces using "real" data to test the system for correctness of response and timeliness of performance.

Step 4: demonstrate recoverability by causing failure in various ways, and test the adequacy of the system to meet recoverability and security requirements.

Step 5: demonstrate robustness by running the Operational Prototype for many hours or days, and test for nondegrading performance in terms of timeliness and correctness of response.

Step 6: demonstrate and test the Operational Prototype running with real data in parallel to the on-line system.

Step 7: run the Operational Prototype on-line and perform final acceptance testing.

Scheduled IPRs should be held at the conclusion of each site installation. Each successful IPR will determine: if the installation is proceeding well, whether to continue with the current step or go on to the next step; if the installation is complete but is not the last installation, whether to start the next installation; or if all installations have been completed, whether to prepare for the Milestone 4 review.

Any major problem arising during an installation step will trigger an unscheduled IPR. In-process reviews may have inauspicious outcomes: terminating the project; changing plans to solve the problem (e.g., by returning to an earlier prototyping phase); or rescoping the application to eliminate the problem. Participants in the IPRs will include project staff, specialists, experts, users and user organization managers, and personnel from the maintenance and operations organizations.

During deployment the various steps and problem descriptions should be documented in the Progress Notebook. All documentation should be updated to reflect the deployed ES. At completion of the final installation all ES products will be updated and delivered to all operational sites (T3).

This chapter is divided into four sections: Section 9.1, this section, is an introduction; Section 9.2 presents an example risk containment table; Section 9.3 describes the management activities; and Section 9.4 describes the technical process.

9.2 RISK CONTAINMENT

Table 9.1 is an example prioritized list of major risk items that need to be identified and addressed during the deployment effort. Since applications vary, the project manager should develop his/her own list of the top ten risk items.

9.3 MANAGEMENT ACTIVITIES

The management activities are organized into:

9.3.1 Joint Management and Technical Task (J1)
9.3.2 Initial Management Activities (M1)
9.3.3 Ongoing Management Activities (M2)
9.3.4 Preparing for the Milestone 4 Management Review (M4)
9.3.5 Holding the Final Milestone 4 Management Review (M5)

9.3.1 Joint Management and Technical Task (J1)

The purpose of this first task to make detailed and explicit deployment plans. This requires the joint participation of the management and technical development teams as well as participation of representatives from the user, operations, and maintenance organizations.

This task specifically addresses: the person at each site to whom the deployed system will be handed; a detailed, phased training and installation plan for deployment at each site; description of the process and terms of the final acceptance testing (what it will be, who will run it, and how problems will be addressed); and a detailed post-deployment plan for development team responsibilities and activities.

Table 9.1

AN EXAMPLE PRIORITIZED LIST OF DEPLOYMENT PHASE RISK ITEMS

	Risk Item	Risk Management Techniques
1.	Personnel shortfalls	Staff this phase with top talent, especially good software manager for software and documentation control and distribution
2.	Unavailability of equipment	Ensure timely delivery of equipment to sites, and ensure availability of AIS(s) and I/O devices; if AIS(s) are used for backup, factor into schedule costs
3.	Lack of site participation	Work continuously with site personnel to ensure (1) user, operator, maintenance training, (2) support and training during installation, (3) availability of site users for testing; appeal to higher management if problems occur
4.	Scheduling coordination problem	Scheduling may be very complicated involving: training, equipment deliveries, and installations; planning with sites ahead of time; making contingency plans in case of problems
5.	Discovery of site-specific needs	Work with site experts in earlier phases (e.g., have them attend IPRs); put off deployment to such a site until post deployment/upgrading
6.	Security accreditation problems	Work out any security problems from previous phase; leave adequate time in schedule for DoD agency to accredit/certify ES before it is operated on-line
7.	Failure to update document products and training products on time	Make it clear to the staff which document and training products they are responsible for, and plan enough money and time for them to update products after each site deployment; have good automated support for document preparation if possible; PM should oversee schedule and review documents; ensure document validation
8.	Ineffective bug report, ECO/ECR process	Plan in advance how to handle minor bugs and changes that require ECOs/ECRs; acquire additional staff if necessary; set up process that can carry over and be used in post deployment
9.	Unrealistic schedule and expectations	Be realistic in plan and schedule; carefully design installation and test steps; recognize when cost and schedule will be inadequate to meet the needs; recognize and handle problems when they occur
10.	Effect of changes in requirements on validity of problem to user mission	Evaluate mission needs whenever considering major changes to problem, integration, security, or communications

9.3.2 Initial Management Activities (M1)

The initial management activities are: (1) to develop a management plan; (2) to ensure that the project has adequate and trained staff personnel (including specialists, user participants, and maintenance and operating personnel); (3) to ensure that the equipment has been received or is in process and on schedule; (4) to carry out plans developed for QA, configuration management, security, and maintenance; and (5) to define the documentation requirements for the deployment effort.

Management Plan. A management plan should include: technical plan milestones for installation by site; schedule for acquiring staff; schedule for acquiring equipment; training schedule (by site for users, operators, and maintenance personnel); maintenance schedule; IPR schedule; and the schedule of monthly meetings to review progress and risk management.

Staffing and Training. It is important that the operational and maintenance people who will support the deployed system participate in installation. It is particularly important to identify the KBAs and to involve them in the process. A major concern during deployment is how to handle changes to the baseline (ECOs, version control, releases, etc.). It is important to train all participants in the deployment process as well as users, operators, and maintenance people.

Equipment. Deployment cannot proceed unless the equipment delivery is in step with the training and deployment schedules. This is a major risk in this phase, and the PM should keep close watch on equipment acquisition. Equipment includes not only the ES tool and workstations but also special I/O devices, off-the-shelf or customized hardware/firmware/software interfaces, and test equipment.

Carrying Out Plans. The PM must ensure the execution of plans developed for QA, configuration management, security, and maintenance.

- The PM should ensure that all technical staff are informed of and/or trained in QA guidelines and configuration management procedures.
- The PM should ensure that the project librarian has training and/or computer aids to conduct the configuration management functions.
- If the deployed ES is to be a secure system, the PM must ensure the enforcement of security measures during the deployment period covering: physical environment (including workstations and disks); clearances for staff members, experts, users, operators, maintenance personnel, and specialists; labeling of inputs and outputs; user/developer access and authorization procedures; etc. For secure systems, the PM must have the Operational Prototype certified or accredited by the responsible DoD agency before the ES goes on-line.
- The PM must plan for: ES tool/workstation installation and checkout; ongoing maintenance of all equipment; and vendor, contractor, or in-house support for the ES tool and/or workstations.

Documentation Requirements. The PM should oversee the updating of document and course products so that the final products reflect the deployed ES. There are no new documents required during this phase. Again, the Progress Notebook should be maintained, and sections should be designated for: the project management plan; the project technical plan; bug reports and solution; ECRs (these replace problem descriptions and solutions); ECOs; and reports on IPRs and monthly management meetings.

The PM should designate a project librarian to be responsible for maintaining the documentation as well as configuration management.

9.3.3 Ongoing Management Activities (M2)

Ongoing management activities include: conducting the project plan for training, installation, testing, and updating documents and courses; monitoring QA, configuration management, T&E, security, and maintenance; facilitating exchanges across the developer,

user, operations maintenance, ADP, security, and communications communities; and planning for post deployment.

Managing the Project Plan. The PM should be sensitive to the risks that may jeopardize the plan. Those that require constant oversight are: equipment deliveries; availability of off-line AISs for use in installation and testing; coordination of training and installation schedules (e.g., being sure there are trained end-users at an installation site to participate in the installation steps); and cooperation and availability of people in the organization who will be responsible for ES life-cycle support so that transference can occur gracefully. The PM should try to discourage returning to an earlier prototyping step for site-specific problems. If only a single site is involved, for example, it may be best to proceed with deployment and make that site's need a high priority for the first upgrade.

The project plan, schedule, and costs should accommodate easily and inexpensively to small changes. This requires careful management so that deviations from goals, schedule, and cost are quickly recognized and addressed without imposing excess management demands on the technical team. The two review techniques for achieving this are: informal monthly project reviews, and more formal, in-process reviews (IPRs).

The monthly reviews should concentrate on progress, risk management, and problems that occurred during the previous month. The review could be in the form of (1) a short written report, (2) a discussion between the PM and technical leader and perhaps a user, operations or maintenance organization representative, or (3) a meeting of all or selected members of the project staff.

In-process reviews should be scheduled at the completion of each site installation. The scheduled IPR should review the completed site deployment T&E results; any unresolved ES or interface problems; any outstanding user, operations, or maintenance concerns; risks; and any changes to the plan. The result of the scheduled IPR may be to proceed with the next site installation (if there is another) or prepare for the Milestone 4 review if all installations are complete. Attendees should include representatives from all interested communities and should, as a minimum, include the project staff, the user organization manager, and site representatives from the user, operator, and maintenance organizations. An unscheduled IPR will be held to address a major problem; attendance should be decided by the PM with recommendations from the technical leader. The outcome of an unscheduled IPR may be to terminate the project or return to an earlier prototyping iteration.

Activities that Need Monitoring. In addition to monthly reviews and IPRs, the PM should:

- Direct the technical leader in monitoring QA.
- Monitor the configuration management activity and address any management problems that occur (e.g., technical staff evading the process).
- Monitor the T&E activity, addressing any problems with: the plan; acquisition of real data cases; participation of the user community or specialists in ADP, security, and communications; addition of real data cases to the T&E database and follow-up on technical response to results.
- Monitor all training activities by talking with people who have been trained and their managers to determine if training is being done well and is sufficient. If not, the PM should see that the courses and materials are modified to be more responsive to needs.
- Monitor risks through monthly meetings. Be particularly aware of risks associated with equipment delivery slippage or unavailability of off-line AIS(s).
- Monitor security policy and methods to ensure proper enforcement during deployment. The PM must arrange for each installation's accreditation or certification by the responsible DoD agency before it can be operated on-line.
- Monitor project support needs: maintenance of tools and equipment, office and laboratory space and usage, secretarial and documentation support, etc.

Facilitate Exchange Across Communities. The PM should, in general, be facilitating information exchange across user, developer, maintenance, operations, ADP, security, and communications communities by keeping them informed and inviting them to relevant IPRs.

In particular, the PM must establish close ties with the user organizations at each deployment site. The PM has a responsibility to keep the site user-organization manager informed as to training, installation, and testing schedules. Close coordination is necessary to get equipment delivered and installed in a timely fashion, to ensure the use of off-line AIS(s), and to ensure mission performance is minimally interrupted by training courses. The PM should frequently discuss user satisfaction with the site user-organization managers, experts, and users.

The PM should continue to work with the maintenance community to develop an understanding of how maintenance will be carried out in the Post-Deployment Phase. This includes which levels of maintenance will be furnished on-site or through local vendors and which will be handled in a more centralized way. This is very dependent on the geographic dispersal of sites and users and the variability of their needs. Although these issues were considered in earlier plans, they may change as a result of site installation experience.

The PM should continue to brief the project to higher management to attract champions and generate enthusiasm and support. Use of a videotape presentation to demonstrate the ES prototype is recommended.

Planning for Post Deployment. The PM should continue to work with the user, operations, and maintenance communities on improvements to the post-deployment plan as a result of site installation experience.

9.3.4 Preparing for the Milestone 4 Management Review (M4)

The main purpose of the Milestone 4 review is to ascertain that the ES installations have been successfully carried out and meet the mission needs of the user organization. Successful outcome of the review would be formal acceptance of the deployed ES and continuation with the Post-Deployment Phase.

Preparation for the Milestone 4 review includes:

- A report on all ES site installations (training, acceptance testing, updates to products and transference to the party responsible for operating the ES).
- Estimated life-cycle costs and the cost benefit/ROI.
- A detailed plan for post-deployment activities.

Report on All ES Site Installations. The PM should prepare an overview of each site installation discussing problems and their solutions and any resulting site-specific needs that may have an impact on life-cycle maintenance. The PM should review the training courses and materials and present an evaluation of their effectiveness based on interviews and/or questionnaires filled out by users, operators, and maintenance people who took the training courses. The PM should ensure that all products have been updated to reflect the deployed ES. The PM should report on the transference to the parties responsible for maintaining the ES and discuss any reservations about the success of the endeavor during the ES life cycle.

Update on Estimated Life-Cycle Costs and Cost Benefit/ROI. The PM should refine the cost benefit/ROI presented at the Milestone 3 review based on experiences gained during the Deployment Phase. Examples of factors that could affect cost estimates are: site-specific needs that will require additional life-cycle costs, underestimated training costs, and unanticipated problems with hardware that will continue into life-cycle maintenance.

Detailed Plan for Carrying out the Post-Deployment Phase. The PM should present a detailed plan of the development project activities in the Post-Deployment Phase. As a minimum the plan should allow for some ES support over a 6-month period, followed by a period to evaluate the ES from user, operator, and maintenance perspectives; it should also include time to write a report. The report should address: validation of ES to user mission and how well it meets the original goals; an update of cost benefit/ROI and projected life-cycle cost; lessons learned; and recommendations (if appropriate) for the first ES upgrade.

The Post-Deployment Plan could include a plan for the first upgrade to be carried out by the development project after the evaluation.

9.3.5 Holding the Milestone 4 Management Review (M5)

The last management activity of the deployment effort is to present the project at the Milestone 4 Management Review. Managers at all levels of the participating organizations (especially those representing user sites) should be encouraged to attend, as well as expert and end-user representatives, representatives from the operations and maintenance organizations, and specialists in ADP, security, and communications. A videotape may be an effective way of describing the deployed ES.

The result of the Milestone 4 review will be (1) formal acceptance of the deployed ES and affirmation of continuation to the Post-Deployment Phase, or (2) a recommendation to return to the Testbed or Operational Prototype to correct recognized inadequacies of the ES, or (3) termination of the project.

9.4 TECHNICAL PROCESS

The technical process of the Deployment Phase can be organized into the following tasks:

9.4.1 Joint Management and Technical Task (J1)
9.4.2 Technical Plan for Deployment (T1)
9.4.3 Installation Steps (T2)
9.4.4 Scheduled and Unscheduled IPRs (M3B, M3A)
9.4.5 Update and Delivery of Final ES Products (T3)

9.4.1 Joint Management and Technical Task (J1)

This task is discussed in Section 9.3.1.

9.4.2 Technical Plan for Deployment (T1)

A detailed technical plan for deployment must be developed that shows the dependencies among training schedules, equipment deliveries and installations, and ES installation by site and across sites. It should indicate personnel assignments to each activity and contingency plans if things go wrong. The plan must include adequate time and money for updating documents, training courses, and training materials after the final installation to reflect all changes to the ES. Note that support for site-specific needs may require support for site-specific documents and training materials.

Each installation will be conducted in accord with the IP prepared in the previous phase. Section 9.4.3 below describes seven incremental steps in an example ES installation. For each step the plan should point to relevant IP sections, T&E materials and procedures, and the estimated cost and schedule to complete the step. An IPR will be scheduled at the completion of each site installation; unscheduled IPRs will be held whenever a major problem arises.

The cost and schedule to carry out the deployment plan should be realistic. If either the cost or schedule is fixed and inadequate to accomplish the work, the technical leader should so indicate and the issue should be resolved before any effort takes place. If the number of installations has to be reduced in order to meet the cost and schedule constraints, then the new deployment plan should be re-evaluated to be sure it still meets user organization mission needs. If it does not, the PM should consider project termination.

9.4.3 Installation Steps (T2)

Each site installation should be carried out according to plan. Before the first installation step, training of operations and maintenance personnel and users should begin. If large numbers of users are at the site and they will be introduced to the new ES in phases,

training classes may be held continuously during ES installation and even after it is completed. Valuable suggestions and comments from the students should be incorporated as updates to the training courses and materials as well as possible system updates.

Selected operations and maintenance personnel who will be responsible for the deployed ES should be trained early on so that they can participate in the installation process. Their comments and feedback should also be incorporated as improvements to the installation procedures and as updates to the document products, as well as possible system updates.

At the completion of a site installation, all ECO changes and their corresponding document updates should be made, with site-specific changes clearly marked. As sites come on-line and require ECO and document changes, new versions of software releases and documents should be distributed to all operational sites. The configuration manager will be responsible for the control and distribution of all software releases and the accompanying documentation until the last installation is completed. At that time, the control and maintenance of all ES products will be handed over to the responsible support organization.

The steps for accomplishing an installation may be dependent on the application and should be described in detail in the IP. For illustrative purposes we have described a seven-step installation process. The first six steps correspond to the six steps used in the Operational Prototype development example described in Chapter 8. The seventh step is to put the ES on-line.

The seven installation steps are:

- Step 1: establish and test low-level communication interfaces
- Step 2: test high-level communication interfaces using test cases
- Step 3: demonstrate and test high-level communication interfaces using "real" data
- Step 4: demonstrate recoverability
- Step 5: demonstrate robustness
- Step 6: demonstrate and test the Operational Prototype running with real data in parallel with the on-line system
- Step 7: run the Operational Prototype on-line and perform final acceptance testing

Steps 1-6 assume availability of an off-line AIS. This could be a backup system used in case of failure or used for testing new releases. An off-line system must be available unless (1) integration is loose (the Operational Prototype only reads from the AIS), and (2) risk is acceptable of causing a failure to the AIS and interrupting its service. An off-line system that is used for backup will not be available for dedicated project use, and possible interruptions to the installation schedule should be taken into account during planning. Note that at least steps 2-6 involve the collection and analysis of runtime information in order to demonstrate or validate that the system is operating correctly or within timing constraints. The collection of the runtime information, in many cases, will exert an effect on the system performance and needs to be factored into the analysis.

Problems found during any of the steps should be documented as bug reports (for minor errors) or as engineering change requests (ECRs). Any minor problems that can be rectified within cost and schedule may be corrected by means of bug reports or the established ECR/ECO process; any major problem will require an unscheduled IPR. Changes made to any step require repeating the T&E for earlier steps in order to ensure that the change does not cause a different problem.

Step 1: Install Equipment and Establish and Test Low-Level Connections. In this step the low-level communication interface between the Operational Prototype and the AIS is implemented using the off-line AIS (or the low-level interface between a special I/O device and the Operational Prototype could be implemented). The interface should demonstrate that: correct data packets can be sent from the AIS and received by the Operational Prototype, performance and error handling is acceptable, different-sized data requests can be handled correctly (i.e., test from the extremes of 1 byte to megabytes), different data flow rates can be handled correctly (i.e., test for errors that would flood the network), and

performance is within acceptable bounds (no unexplained delays). The same characteristics should be demonstrated for data packets being sent from the Operational Prototype to the AIS. Finally, robustness should be tested by (1) exchanging packets for long time periods and (2) causing failure of the network, AIS, and Operational Prototype and evaluating recovery characteristics.

Test and evaluation should be carried out at the completion of this step. Any minor problems that can be corrected within cost and schedule should be corrected in accordance with ECOs. Major problems are unexpected unless the site is using a different I/O device or network interface from those used by the prototype site. If so, and there is a major problem, an unscheduled IPR should be held. Successful conclusion of this step demonstrates that low-level communication between the Operational Prototype and AIS works.

Step 2: Demonstrate and Test High-Level Connections Using Test Cases. After step 1 has passed T&E, the high-level Operational Prototype functions can be demonstrated using the test cases described in the IP. The test cases or test case database should be put on the off-line AIS in order to demonstrate that it can be correctly read from the AIS to the Operational Prototype. It should be processed on the Operational Prototype and the answers verified. The updates or new data output from the processing should be sent to the AIS and verified there. If the criteria include timing requirements, then timing information should be collected and analyzed on the AIS, network, and Operational Prototype. Proper handling of system security (e.g., physical security; user authorization; marking of classified output screens and printed information; and data, computer, and system security) should be confirmed.

Test and evaluation should be carried out at the end of this step, and there should be no insolvable problems. Successful T&E demonstrates that the Operational Prototype handles the interface and test cases correctly and within timing constraints and that security requirements have been met.

Step 3: Demonstrate High-Level Connections Using "Real" Data. To test the system for correctness of response and timeliness of performance, step 3 uses the communication interface demonstrated in step 2 to test how well the Operational Prototype handles the site's "real" data. Captured "real" data from the on-line system is entered on the off-line AIS to be read by the Operational Prototype. The "real" data will need to be evaluated by experts as part of T&E, and criteria will need to be established in order to determine if data are being properly processed by the Operational Prototype.

Test and evaluation should be carried out at the completion of this step. As part of the T&E, local end-users who have processed the "real" data cases in the current on-line system should process the same "real" data cases using the Operational Prototype. Their evaluations will help determine how well the Operational Prototype (especially the user interface) meets their needs; it is important to resist gold plating. End-user evaluations should affect the process of correcting problems not that of adding functionality. Suggestions for additional functionality should be written up in the Progress Notebook where they can be considered in post-deployment upgrades.

It is possible that problems may occur at this point. Examples are: a site-specific need for data/knowledge handling may be discovered; the Operational Prototype may incorrectly handle a "real" data example that did not come up in previous tests; or a category of data may have been overlooked during problem definition. Any minor problem that can be corrected within cost and schedule should be written up in a bug report or follow the established ECR/ECO process. Major problems (e.g., correcting the processing of a particular case category) will require an unscheduled IPR.

Successful conclusion of this step demonstrates that the Operational Prototype handles "real" site data correctly. The "real" site data should be added to the IP T&E test database. This may be done selectively, adding only those new cases that caused a problem or that differ in a significant way from existing cases.

Step 4: Demonstrate Recoverability. Testing the adequacy of the system to meet recoverability and security requirements should be demonstrated by: causing a power

outage; attempting to violate security policy and mechanisms; pulling plugs on the Operational Prototype disks, workstation, network connection, and AIS; causing probable errors to occur with the data themselves (e.g., wrong data values, missing data, duplicate data); and causing failures in sending and re-sending messages (e.g., too many messages or too many re-sent messages, lack of network response).

Test and evaluation should be carried out at the end of this step. Any minor problems that can be corrected within cost and schedule should be written up in a bug report or follow the established ECR/ECO process. If the same equipment is used as in the previous phase, it is unlikely that a major problem will occur. If a major recovery problem does occur, an unscheduled IPR should be held to address it. Successful T&E demonstrates that the Operational Prototype can recover from the failures it was tested for.

Step 5: Demonstrate Robustness. The next step is to demonstrate robustness. This example assumes that the ES and AIS will be running continuously. This step must demonstrate that the Operational Prototype and AIS can run continuously for a test period of, for example, 7 days without performance degradation (i.e., timeliness and correctness of response) and in cases of failure can recover as previously demonstrated. This will require having end-users operate the system.

Runtime information will be collected, and the collection process will affect the runtime performance and thus needs to be taken into consideration when analyzing performance. It is important to make sure that the data cases are extremely varied. If possible, "real" data cases should be captured and used. If the detailed analysis of 7 days' worth of such data is too costly, statistical methods can be used to select meaningful samples to analyze in detail. Graphs of performance over the runtime period can help identify possible trouble periods that will require a more detailed examination.

Test and evaluation should be carried out at the completion of this step. Any minor problems that can be corrected within cost and schedule should be written up in a bug report or follow the established ECR/ECO process. Major problems will require an unscheduled IPR. Successful conclusion of this step demonstrates that the Operational Prototype and off-line AIS can successfully sustain operation over the stated test period, thus demonstrating system robustness.

Step 6: Demonstrate and Test the Operational Prototype Running with Real Data in Parallel with the On-line System. The goal of this step is to run the Operational Prototype in parallel with the existing on-line system. To do this in a data-driven Operational Prototype, one needs to capture and port the data from the on-line system to the prototype system in real time or close to real time. If the system is user-driven, true parallelism may be difficult or impossible to achieve. The best that can be done might be to give the users of the prototype the same tasks that the on-line users were given.

This step should be conducted in two parts. The first part should have runtime performance collection turned on in order to demonstrate that the Operational Prototype can meet or come close to meeting performance criteria. The second part should be run with runtime collection turned off or removed so that the Operational Prototype runs as it will when it goes on-line.

Test and evaluation should be carried out at the completion of this step. In particular the end-users should do a final evaluation of the user interface, performance, and support features. Any minor problems that can be corrected within cost and schedule should be written up in a bug report or follow the established ECR/ECO process. Major problems (e.g., change to the security mechanism, added functionality, change to user interface) will require an unscheduled IPR. If the ES is a secure system, it must be accredited or certified by the responsible DoD agency at this time in order to go on-line in the final installation step. Successful conclusion of this step means that the Operational Prototype installation in off-line mode is completed.

Step 7: Run the Operational Prototype On-line and Perform Final Acceptance Testing. The final step in the installation process is to put the ES on-line and turn off the old system. There should, as with all AISs, be a well-defined backup mode in case of system failure.

The final acceptance tests will be made by the organization responsible for the deployed ES. The technical team and all specialists and experts should be available during the acceptance period.

9.4.4 Scheduled and Unscheduled IPRs (M3B, M3A)

During this phase scheduled IPRs will be held at the completion of each site installation. An unscheduled IPR is held when a major problem is identified that may cause considerable change to the plans and schedule. Unscheduled IPRs may be called any time during deployment.

The people who should attend the IPR include: all project staff and managers; the user organization managers (in particular the user site manager); site maintenance and operations personnel who will be supporting the deployed ES; experts and end-users who have participated in the installation; and specialists in ADP, communications, and security who have participated in the installation.

Unscheduled IPR. The unscheduled IPR should address the current status of deployment, the ECR that prompted the IPR, alternatives for problem solution, and recommendations from the technical team. The PM (with recommendations from the KE) should decide whom to invite to the unscheduled IPR.

What to do about the problem will depend on a number of factors:

- How well the deployment process is proceeding.
- How important ECR implementation is to the user community and its mission. If the problem is worth solving then:
  - Availability of additional funding,
  - Acceptability of lengthening the schedule and/or finding additional staff.
- The amount of risk involved in straining technology (ES, ADP, security, telecommunications, etc.).
- The degree of "gold plating" (capabilities outside the defined problem scope) required by the solution of the problem.

Alternative outcomes of the IPR are:

- To rescope the application to eliminate the need to implement the ECR at this time, to change relevant plans, and to continue deployment.
- To obtain additional funding and a schedule extension and/or additional staff to address the ECR, to change relevant plans, and to go back to an earlier prototyping phase if change is extensive and system cannot be deployed without it.
- To terminate the project.

The IPR proceedings and outcome should be reported in the Progress Notebook.

Scheduled IPR. The scheduled IPR should include:

- A review of deployment progress to date, and a review of all major and minor bug reports, ECRs, and ECOs addressed since the previous IPR.
- A review of all T&E results and any shortcomings.
- A review of any ECR uncovered by the IPR to be handled as described under "Unscheduled IPR."

Possible outcomes of the scheduled IPR are:

- To accept the ES as deployed at the operational site and transfer the deployed system from the development team to the organization that will be supporting it.
  - If this is not the final installation, then proceed with the next installation.
  - If this is the final installation, then update the final ES products and prepare for the Milestone 4 review.

The IPR proceedings and outcome should be reported in the Progress Notebook.

9.4.5 Update and Delivery of Final ES Products (T3)

At the completion of the final installation, all ES products should be updated and delivered to all operational sites. This includes all documents described in Section 8.4.5 and training courses and training materials. Updates should include special identification of site-specific information.

A report on the deployment experiences and the review of the final products should be input to the management preparation for the Milestone 4 review.

10. POST-DEPLOYMENT PHASE[8]

10.1 INTRODUCTION

Figure 10.1 shows the place in the expert system development process of the Post-Deployment Phase. Its major technical and management activities are shown in Figure 10.2. During the Post-Deployment Phase the ES will be operated and maintained by the ES support organization to which it was transferred at the end of the Deployment Phase. The main project responsibilities during post deployment are to: help the ES support organization carry out its mission; evaluate the ES during its first 6 months of operation; update the ES life-cycle and ROI cost estimates; and participate in the ES upgrade plan and strategy. Though the activities are shown as spanning the first year of post deployment, this is meant as an example. The exact time frame for performing these tasks will depend on the ES and the needs of the ES support organization.

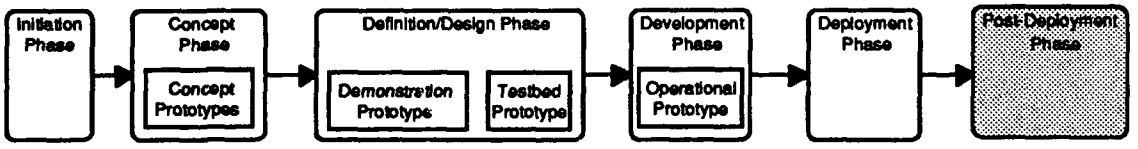


Figure 10.1—Expert system development process, Post-Deployment Phase

An important outcome of the first 9 months is an ES upgrade plan and strategy to present to management review for approval and support. If substantial ES upgrades are needed, continuity of development would be best assured by having the team that developed the ES perform the upgrade. The first upgrade would have to begin immediately after the first post-deployment year in order to have the project team available and committed to the effort.

Overview of Management Activities. The first activity in this phase is to review the post-deployment plans and develop a detailed plan for how the project will participate in the Post-Deployment Phase (J1). This requires joint participation of the management and technical teams as well as participation of representatives from the user and ES support organizations.

After detailed plan agreement, the project manager's (PM's) initial management activities (M1) are (1) to ensure that the project has the trained staff and necessary equipment to carry out the plan and (2) to define the documentation needs for this phase. Ongoing management activities (M2) include: seeing that the project's part of the post-deployment plan is carried out; negotiating with the ES support organization regarding any modifications to the plan; continuing to work closely with the user and support organizations for feedback on how well the project is meeting its obligations; and exploring the role of the project team in future upgrade activities.

The PM will hold an IPR at the end of the first 7 months to review the project's support effort and the ES evaluation information, lessons learned, and recommendations (M3). During the following 2 months the evaluation information will be used by the development team, user organization, and ES support organization to develop an upgrade plan and strategy. At the end of 9 months, the PM will participate in a management review of the upgrade plan and strategy. If the outcome of that review is management support and funding

[8]See Appendix A for definitions of acronyms and terms used in this chapter.

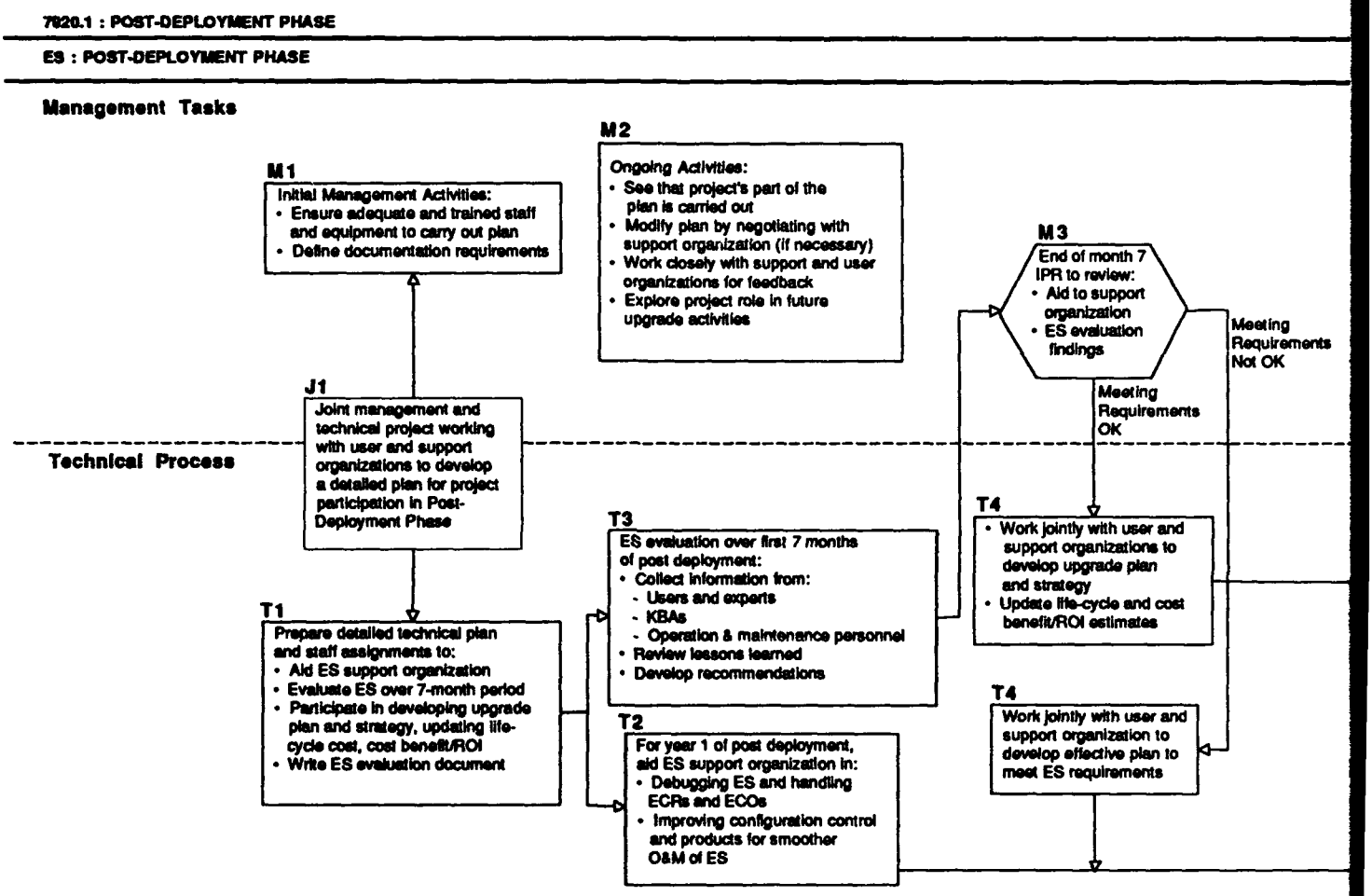
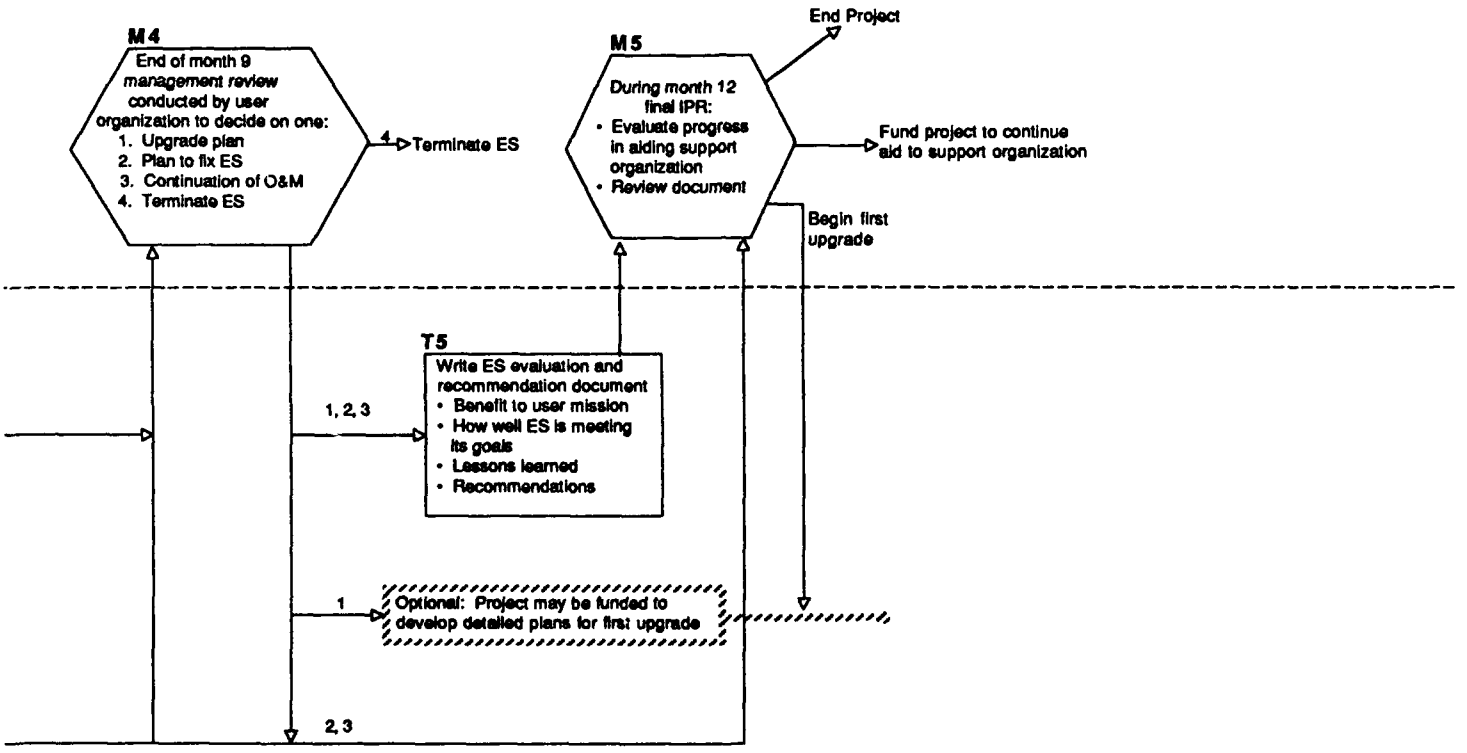


Figure 10.2—Post-Deployment Phase


M4
End of month 9
management review
conducted by user
organization to decide on one:
1. Upgrade plan
2. Plan to fix ES
3. Continuation of O&M
4. Terminate ES
4
Terminate ES
M5
During month 12
final IPR:
• Evaluate progress
in aiding support
organization
• Review document
End Project
Fund project to continue
aid to support organization
Begin first
upgrade
T5
Write ES evaluation and
recommendation document
• Benefit to user mission
• How well ES is meeting
its goals
• Lessons learned
• Recommendations
1, 2, 3
1
Optional: Project may be funded to
develop detailed plans for first upgrade
2, 3

for a near-term ES upgrade to be performed by the project, the PM and technical team will develop a plan for the first upgrade prototype effort to begin the following year. A final IPR will be held at the end of the year (1) to evaluate the progress made by the project in aiding the support organization and (2) to review all required documentation. If the project is not to be involved in an upgrade effort, a decision may be made at that time to extend the project's role in post-deployment activities.

Overview of Technical Tasks. After concurrence on the detailed plan for project participation in the post-deployment effort, a detailed technical plan (T1) should be prepared. This plan will indicate staff assignments to the various activities: aiding the ES support organization in maintaining the ES; evaluating the ES over the first 6-month period; participating in developing ES upgrade plans and strategy; updating estimates to the life-cycle cost and cost benefit/ROI; and writing the evaluation document.

The technical team will be available to aid the ES support organization throughout the year in maintaining the system and making minor enhancements (T2). Its presence over this period should help ease the transition from deployment to post deployment. It will aid in: debugging; handling ECRs and ECOs; and improving or making smoother the configuration control of software, documents, and training courses.

During the first 6 months the technical team will evaluate the ES from the perspectives of user mission, users and experts, KBAs, operators, and maintenance personnel (T3). The evaluation will include lessons learned and recommendations that will be presented at an IPR at the end of the seventh month. These results will be used by a joint project team, user organization, and ES support organization effort to develop an upgrade plan and strategy and to update the estimates for life-cycle cost and cost benefit/ROI (T4). The upgrade plans will be acted upon by management at a review held at the end of 9 months. Finally, the evaluation and recommendation document will be written (T5).

This chapter is divided into four sections: Section 10.1, this section, is an introduction; Section 10.2 presents an example risk containment table; Section 10.3 describes the management activities; and Section 10.4 describes the technical process.

10.2 RISK CONTAINMENT

Table 10.1 is an example prioritized list of major risk items that need to be identified and addressed during the post-deployment effort. Since applications vary, the project manager should develop his/her own list of the top ten risk items.

10.3 MANAGEMENT ACTIVITIES

The management activities are organized into:

10.3.1 Joint Management and Technical Task (J1)
10.3.2 Initial Management Activities (M1)
10.3.3 Ongoing Management Activities (M2)
10.3.4 Management Reviews (M3, M4, M5)

10.3.1 Joint Management and Technical Task (J1)

The first activity in this phase is to review the post-deployment plans and develop a detailed plan for how the project staff will participate in the Post-Deployment Phase. This requires the joint participation of members of the management and technical teams as well as participation of representatives from the user and ES support organizations.

Table 10.1

AN EXAMPLE PRIORITIZED LIST OF POST-DEPLOYMENT PHASE RISK ITEMS

	Risk Item	Risk Management Techniques
1.	Personnel shortfalls	Staff this phase with top talent, find special people to perform ES evaluation study
2.	Personnel problems with carrying out joint support activities	Hold joint monthly meetings with user and support organizations to check how well people are working together; PM should frequently check with other personnel especially at remote sites
3.	Lack of user, site, support organization participation	Work with users, remote user sites, and support organization to get commitment and cooperation required to carry out the ES evaluation task; do this on a continual basis during the 6-month effort
4.	Inadequacies in configuration control	Work closely with support organization to detect and suggest changes to configuration control methods to better meet support organization needs
5.	Slippage in cost and schedule in solving bugs and carrying out ECOs	Review specific tasks at monthly meetings with user and support organizations to detect and address problems before they become acute
6.	Lack of staff supervision at remote sites	PM should frequently check on progress at remote sites; PM should talk to user and support organization personnel frequently to be sure needs are being met
7.	Inadequacy of courses, documents, and/or manuals	All staff should be alert to user, KBA, and support personnel difficulties that could be lessened by improving courses, manuals, and documents; suggestions should be passed on to the support organization by the PM and technical leader; aid the support organization in making changes if requested to do so
8.	Unrealistic upgrade plans	Check upgrade plans against actual costs and schedules for building current ES and comparable upgrades of other systems
9.	Unauthorized user changes to ES or ES tool	Though the ES and deployed tool may supposedly not allow unauthorized changes, work with users and support personnel to be sure users understand reasons and process of change control; encourage support personnel to be responsive to user requests
10.	No plan for beginning immediate ES upgrade	Brief management and other potential users on the benefits of beginning the upgrade while there still is a nucleus project with knowledge about the ES development effort

The plan should address how the tasks to be carried out by the technical team will be managed in cooperation with the user and support organizations. The main technical team tasks are to: aid the ES support organization, perform the ES evaluation study, participate

in the upgrade plan and strategy, update cost and benefit estimates, and write the evaluation and recommendation document.

The main technical team issue with respect to aiding the support organization is how technical team assignments are to be handled. For example, one way of handling assignments might be to form a small management group (with representatives from the project, user, and support organizations). They would have a charter (1) to review program bugs, ECRs, and ECOs with regard to how urgent they are, their cost estimates, the technical expertise required, the availability of technical expertise, their priority ranking, etc., and (2) to make decisions about what should be done in various versions and releases of the ES, when, and by whom. The supervisors of the people assigned to the tasks would be responsible to the committee for seeing that the work is done.

In order for the technical team to carry out the ES evaluation, the user and support organizations will have to be cooperative regarding providing people, reports, and perhaps some interactive sessions to explain the problems. Issues include how to support the evaluation process (through planned evaluation-gathering sessions, in-depth interviews of selected people, questionnaires sent to many people, etc.), and who has responsibility if, for example, interviewees are not available or cooperative or questionnaires are not returned.

Joint efforts are required for developing the upgrade plan and strategy and updating the cost and benefit estimates. Issues to be considered include how these efforts will be managed, who will have primary responsibility for them, and how the participants are selected.

10.3.2 Initial Management Activities (M1)

After the project plan has been jointly developed, other initial management activities are (1) to ensure that the project has the trained staff and necessary equipment to carry out the plan and (2) to define the documentation they will be responsible for.

The composition of the project staff will change from deployment to post-deployment. For example, in post deployment fewer programmers may be required because programming skills may be available through the support organization, but new staff may be needed to evaluate the ES. Any new staff will have to be trained. In addition, the PM needs to ensure that all staff members have the necessary equipment to support them in performing their jobs.

Last, the PM should clearly specify the documents required during this phase and ensure that the management plan allocates enough time and funds to produce them. At least one new document, the ES evaluation and recommendation report, will be written by the technical team, but the plan could call for additional documents (e.g., strategy for upgrades, or a detailed plan for the first upgrade). The PM should designate a project librarian who will be responsible for maintaining the documentation.

10.3.3 Ongoing Management Activities (M2)

Ongoing management activities in post deployment include: ensuring that the technical team effort is carried out; negotiating with the ES support organization over any modifications to the plan; continuing to work closely with the user and support organizations for feedback on how well the technical team is meeting its obligations; and exploring the role of the project in future upgrade activities.

Managing the Technical Team Effort. The PM should be sensitive to the risks that may jeopardize the plan. Some risks that require constant oversight are: joint tasks that require coordination of people from different organizations; debugging problems and/or engineering changes that exceed cost and schedule estimates; and lack of well-supported and enforced configuration control techniques. Another problem to be considered is coordination and supervision of project staff across different sites.

Negotiating Plan Modifications. As with the other development phases, the project plan, schedule, and costs should accommodate easily to small changes. Larger modifications to the plan may result from user and/or support organization requests or from technical

problems (e.g., bugs that are difficult to eliminate, bad estimates). Since the plan was jointly negotiated, all significant changes will also need to be negotiated. This requires careful management of the process so that problems and deviations from goals, schedule, and cost are quickly recognized and addressed. The two review techniques for achieving this are: informal monthly project reviews, and more formal, in-process reviews (IPRs).

The monthly reviews should concentrate on progress, risk management, proposed modifications to the plan, and problems that occurred during the previous month. The PM and at least one representative each from the user and support organizations should meet for this review, as well as people directly concerned with any problem being addressed.

Working Closely with the User and Support Organizations. The PM should work closely with the user and support organizations at the various sites to obtain feedback, since the outlying sites may be unable to send representatives to the monthly meetings. The PM concerns should be: (1) how well the ES is functioning; (2) how well the training courses and documents are meeting the needs of users and support people; and (3) how well the technical team is carrying out its support activities.

The PM should brief the degree of ES success to higher management to attract champions and generate enthusiasm and support for upgrades. Use of a videotape presentation is recommended.

Exploring the Role of the Development Team in Future Upgrade Activities. The PM should participate in the discussions about future upgrade activities and point out the risks if these activities dc not begin before the experienced project staff disperses.

10.3.4 Management Reviews (M3, M4, M5)

Three formal project reviews will be held during the first year: an IPR at the end of the seventh month, a management review at the end of the ninth month, and a final IPR during the twelfth month.

An IPR will be held at the end of the seventh month (1) to review the progress of the technical team in aiding the support organization and (2) to review the ES evaluation effort, lessons learned, and recommendations. The people who should attend include all project staff and managers; user and support organization managers and/or representatives from all sites; and specialists in ADP, communications, and security who have been involved with post-deployment. In addition, user organization representatives not currently using the ES but planning to do so after an upgrade may want to attend the ES evaluation review. The reason for having the IPR at the end of the seventh month is to allow a month for preparation after completing the ES evaluation study.

Possible IPR outcomes with respect to project aid to the support organization include: (1) if the project is performing well, to continue with the current plan; or (2) if the project is not performing well or the job is not getting done, to make recommendations for a more effective plan and to suggest an IPR or a written progress report at a future date to follow up on the recommendations.

Possible IPR outcomes with respect to the ES evaluation are: (1) if the evaluation is not positive, then the project, user, and support organizations should jointly prepare a plan (including schedule and cost) for meeting the ES functional goals and present the plan at the management review at the end of month 9; or (2) if the evaluation is positive, then the project, user, and support organizations should jointly prepare an upgrade plan and strategy and present the plan at the management review at the end of month 9.

The management review held at the end of the ninth month should be conducted by the user organization and attended by the project and support organization representatives. A plan should be presented either for upgrading the ES to meet its stated functional and performance requirements or for upgrading the ES to satisfy an additional set of functional requirements. Possible outcomes may be (1) to support a near-term upgrade to be carried out by the project in the following year, (2) to support an upgrade plan to be carried out further in the future, (3) to reject the upgrade plan offered but continue ES support and maintenance, or (4) to terminate the ES.

If the ES was not terminated after the management review, the final IPR of the project will be held during the twelfth month (1) to evaluate the progress made by the project in aiding the support organization and (2) to review all required documentation. If the project is not to be involved in an upgrade effort, a decision may be made to extend the project's role in post-deployment activities.

10.4 TECHNICAL PROCESS

The technical process of the Deployment Phase can be organized into the following tasks:

10.4.1 Joint Management and Technical Task (J1)
10.4.2 Technical Plan for Post Deployment (T1)
10.4.3 Project Aid to Support Organization (T2)
10.4.4 Evaluation Study of ES (T3)
10.4.5 Development of Upgrade Plan and Cost Updates (T4)
10.4.6 Documentation (T5)

10.4.1 Joint Management and Technical Task (J1)

This task is discussed in Section 10.3.1.

10.4.2 Technical Plan for Post Deployment (T1)

A detailed technical plan for post deployment should be developed that indicates staff assignments to the various activities: aiding the support organization in maintaining the ES; evaluating the ES over the first 6-month period; participating in developing the ES upgrade plans and strategy; updating life-cycle and cost benefit/ROI estimates; and writing the evaluation document. Two activity threads will be going on in parallel: (1) aid to the support organization and (2) ES evaluation. The latter will be followed by development of an upgrade plan, updated cost estimates, and documentation. Different staff may be required for each set of activities, though senior design and development staff will participate in both. Considerations for planning include:

- Coordination of the technical team with the support organization.
- Location of staff at geographically remote user sites.
- Supervision of junior staff at remote sites.
- Coordination of users and support people for the evaluation study.
- Whether the ES evaluation will include all sites equally or mainly and intensively address a single site.

10.4.3 Project Aid to Support Organization (T2)

The project will aid the support organization throughout the first year in debugging, handling ECRs and ECOs, and making improvements (if necessary) to the configuration control mechanisms to support a smoother operation.

The technical team can help the support organization:

- To understand and fix programming bugs in the ES and, in doing so, help the support people orient themselves better with the program code. As a result of this experience, technical staff may recommend changes to the documents and/or courses.
- To work with users to clarify ECRs. The technical staff may (1) recognize an ECR as a need acknowledged earlier in development but left to be handled in an upgrade, or more seriously (2) may realize that an ECR addresses an aspect of the problem that was never considered before and may require a deeper investigation (e.g., building a new concept prototype).

- To help KBAs make modifications to the knowledge base. Information about the rate and complexity of knowledge base changes should be used to improve the ES life-cycle cost estimate. If the KBAs are newly trained and have difficulty maintaining the knowledge base, relevant changes to documents and courses should be recommended.
- To address ES problems that result from upgrades to the ES tool, operating system, AIS, DBMS, and other support tools. New tests may need to be designed and implemented to detect possible problems.
- To help adjust the configuration management and control techniques used during ES development and transferred over to the support organization at the end of the Deployment Phase. Although well-documented, familiar to the project staff, and included in training courses for the operational staff, these techniques might not fit well into the support organization environment or way of doing business and may need to be adjusted. This may be true if the ES is dispersed to multiple sites and particularly if sites have specific needs. In these cases, the need to simultaneously support multiple versions may require additional configuration control mechanisms.
- To help organize ECOs by determining which change or changes should be addressed in a particular ES version or release. In the case of an ECO addressing a particular site need, a variant version of the ES may have to be created.

10.4.4 Evaluation Study of ES (T3)

The ES evaluation study will be undertaken during the first 6 months of the Post-Deployment Phase. The first 3 to 4 months will be used to organize the study. During that time, decisions will be made with respect to:

- The techniques to be used to acquire the information.
- The sites that will participate in the study.
- A schedule for observing ES users, observing the KBA's activities, and observing the operation organization's activities in maintaining the ES.
- A schedule for interviewing selected individuals.
- The design of questionnaires to collect information from a broader base than can be reached in interviews.

The next 2 months will be spent collecting the information and the following month doing the analysis and preparing a briefing for the seventh month IPR. A document describing the study and recommendations should be completed by the beginning of the twelfth month.

The main objectives of the ES evaluation study are to determine:

- How well the ES meets its goal.
- The ES shortcomings and how serious they are.
- What is being done to improve the shortcomings.
- Recommendations for ES improvements (both near term and long term) that will feed into upgrade plans and strategy.
- Lessons learned from the ES development, deployment, and post-deployment efforts that can be shared with and benefit other ES efforts.

Tora Bikson of The RAND Corporation has spent considerable time investigating the transfer of interactive tools into the user setting and has made important observations about the process. The transfer process itself—how it is conceived and carried out—has much to do with whether and how much ES potential is realized. There should be no sharp distinction between where the ES development process ends and the transfer process begins. Expert System users should be involved in the design of the interface early in the process, and, in the late stages of development, the distinctions between iterative prototype refinement, pilot trial, and initial use should be blurred.

When the ES is declared to be "operational," however, often both developers and users think the most difficult part of the effort is over. In fact, training/learning and support for full utilization take relatively longer and are more labor intensive than either group is usually prepared for. While users are not technically naive, they will not be experienced with the new ES tool and may be inexperienced with expert systems applications development in general. Technical users need a course of formal instruction in addition to manuals and documentation. One-shot training is much less effective than spaced sessions because new users must initially focus on how to operate the workstation, tool, and ES, and only then learn how to use the ES in performing their work. Additional courses should be held when new features are added to the ES or tool in later versions and releases.

Some period of disruption and reduced "productivity" should be expected as users are coming up to speed with the ES, and managers are wise to plan for it. While acquiring knowledge and skill in the use of the ES, users cannot be expected to keep up with regular work loads. Also during this period, it is important to reward users for experimentation and for learning rather than holding them accountable for unmet schedules or mounting backlogs.

Successful completion of the transfer does not mean that the ES features are finalized, that capabilities have been perfected, or that stable, unchanging usage should be expected. Rather, implementation success is marked by continuing reciprocal adaptation of the ES and tasks.

In a successful ES endeavor, as users gain experience with the ES they will develop new approaches to old tasks, do things that were not possible with the old methods, and in general "reinvent" their work. At the same time, users should be encouraged to discover ways to adapt, modify, and extend the ES to better meet their work demands. Such alterations should be regarded as the means by which users' expertise gets coupled with the ES and ES tool for long-term success.

The way in which the user organization supports alterations to the ES and ES tool has important implications for ease or difficulty in maintaining the ES. Adaptations and modifications can be handled in several ways. They can be fed into the established ECR/ECO system, they can be implemented experimentally under support organization control, or they can be implemented directly by the user. The advantages of the former two methods are that they offer control over changes, they are well planned, and they can be maintained by the support organization. The disadvantage of direct changes by the user is that such modifications may be undocumented, may be difficult to maintain, and may cause problems in other parts of the ES that, to the user, appear unrelated to the changes he/she has made. If users share their changes informally with other users, the ES could soon degenerate into many unlike and unsupportable versions. However, some kinds of modifications can be carried out safely by users. For example, an ES may be designed so that some user-interface functions can be modified by the user. Users should be encouraged to customize their ES within these limits.

Rather than trying, probably futilely, to eliminate local talent, user organization managers should be encouraged to invest in and support it as sources of ES-approved customization and technical assistance (even though conventional wisdom may have it that users should not be doing what the support organization is paid and trained to do). The desired result, in the long term, is full incorporation of the new technology's potential in a way that enhances users' substantive expertise and that is adaptive to the work context.

10.4.5 Development of Upgrade Plan and Cost Updates (T4)

The ES upgrade plans should be developed jointly by representatives of the user and support organizations. The upgrade begins with the recommendations made by the ES evaluation study and organizes them into a plan that may encompass more than one upgrade phase.

To develop a meaningful upgrade plan, each new application feature has to be carefully considered with respect to:

- A detailed description of the feature's functionality.
- Risk of feature implementation with respect to straining technology.
- The dependency of the feature on existing ES features and on other recommended upgrade features (and the dependency of existing ES features and recommended upgrade features on it).
- The dependencies of feature implementation on tool, AIS, or operating system capabilities that are not available now but will be available in a future release.
- The feature's estimated cost of implementation, schedule constraints, and anticipated cost benefit/ROI advantages (e.g., the feature may result in an increase in the number of users).

If these questions about a feature cannot be easily answered, further research may have to be performed (e.g., by working with an expert or building a small concept prototype) in order to understand the feature well enough to include it in an upgrade. This can be considered a limited return to the Concept Phase.

Once the questions are answered, the features can be organized into dependency and user priority structures. These can be used to decide which features to group together in each upgrade phase.

Finally, a plan should be developed for each upgrade phase that should include: the features to be developed along with the information collected about them; how the implementation will proceed (e.g., beginning with a Demonstration Prototype effort using the current Testbed Prototype); estimated cost, schedule, and staff required for each phase; and estimated cost benefit/ROI.

Finally, an updated ES life-cycle estimate can be made based on the ES evaluation study results and the upgrade plan.

The upgrade plans and updated costs should be documented in the Progress Notebook or the equivalent being maintained by the support organization.

10.4.6 Documentation (T5)

The minimal documentation required for this phase is a document describing the ES evaluation study effort. It should include:

- How well the ES meets its goal.
- The ES shortcomings and how serious they are.
- What is being done to improve the shortcomings.
- Recommendations for ES improvements (both near term and long term).
- Lessons learned from the ES development, deployment, and post-deployment efforts that can be shared with and benefit other ES efforts.

Appendix A

TUTORIAL AND GLOSSARY

A.1 EXPERT SYSTEM PROGRAMMING PARADIGMS AND REPRESENTATIONS

In this section we discuss several expert system techniques that extend a programmer's capabilities to represent and apply knowledge. Currently there is not a definitive set of knowledge representation paradigms. Most expert system tools use several of these techniques in concert. However, the degree to which they are effectively integrated varies widely. We discuss each of the major paradigms individually and give an example of an expert system tool that exploits this particular technology. Next we cover various characteristics of expert system tools that facilitate knowledge based system development. In addition two glossaries are provided: a glossary of expert system terminology to facilitate the understanding of the artificial intelligence literature, and a glossary of logistic terms taken mainly from the Logistics Management Institute [Young 1987].

A.1.1 Rule Based

In a rule based expert system, domain knowledge is encoded as a collection of *rules* and a knowledge base of facts. Rules often take the *If* **A** *Then* **B** form. Expert system inference engines can process rules in a forward and/or backward manner. In the forward direction, *forward chaining*, **A** represents predicates involving knowledge base facts and **B** represents actions to be carried out if the predicates are true. The action part of these rules can change data values in the knowledge base thus causing other rules to execute.

```
IF:          SPILL MATERIAL IS AN ACID
       AND   SPILL SMELLS LIKE VINEGAR
THEN:        SPILL MATERIAL IS ACETIC ACID
```

Figure A.1—Forward chaining rule

In the backward direction, *backward chaining*, **B** represents a desired result, *goal*, and **A** represents precondition predicates that must be true to support the goal. Thus the precondition predicates become goals that must be supported by either knowledge base facts or other goals. In the following examples the knowledge base contains information about the current state of the house sensors.

The difference between forward and backward chaining of rules is how the induction engine treats the rules. The choice of the rule processing technique to be used is usually statically specified at rule creation. Some systems offer only one style of chaining, e.g., PROLOG [Clocksin 1981] uses only backward chaining. Others allow the user to indicate which direction of processing is currently active, i.e. , forward only or backward only.

Several other concepts are important in understanding rule based expert systems. As rule based systems grow in size, keeping track of which rules participate in which decision components becomes more difficult. To ameliorate this, rules of similar function are clumped together into *rule sets*, a concept similar to subroutines in programming languages. The rule

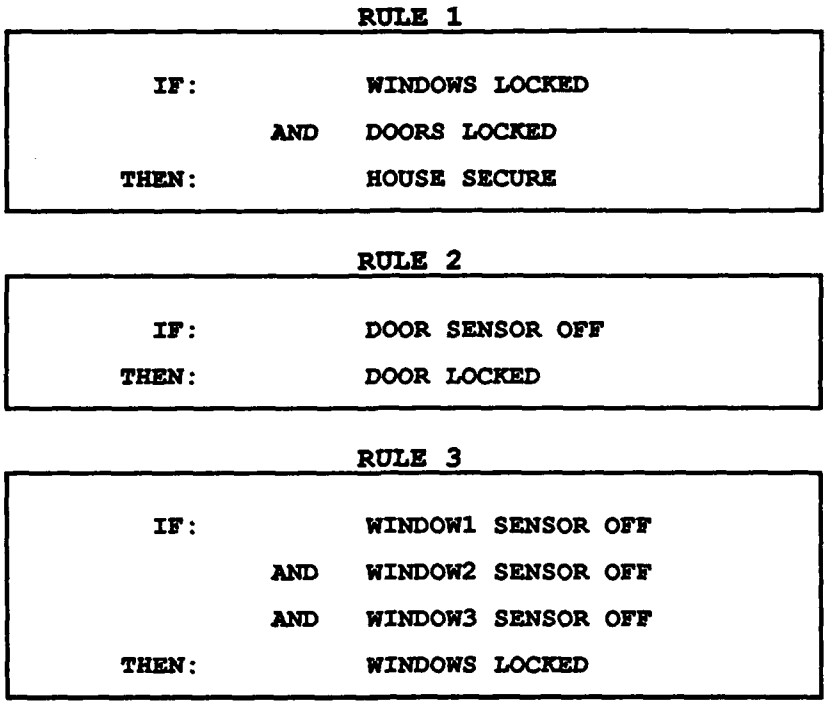


Figure A.2—Backward chaining rules

execution component of an expert system, the *inference engine*, executes *(fires)* instantiated rules. At any time during this process there may be several rules that are ready for execution; they are called the *agenda* or *conflict set*. The scheduling of who is next on the *agenda* of instantiated rules is important to the overall performance of the expert system. Expert system tools differ in the extent to which the user has control over this scheduling process. The representation of the rules in a rule based system also affects the performance. Some systems interpret and some compile the rules. The performance difference for compiled rule based systems can be ten times that of an interpreted system. Many expert system shells offer rule based knowledge representation (e.g., ART™ [Clayton 1985], EMYCIN [Shortliffe 1976], OPS5 [Forgy 1981]).

A.1.2 Semantic Nets

The semantic net knowledge representation model consists of a network of points called *nodes* and arcs called *links*. The nodes in a semantic net represent objects, concepts, or events. The links represent the relationship between the connected nodes. Semantic networks used to describe inheritance hierarchies have links like *has-part* and *isa*. This style of knowledge representation offers several advantages over traditional flat databases. By attaching common properties to more general nodes, the space required to represent multiple similar objects is much less. Also the taxonomic representation provides a convenient separation between the hierarchical description of a concept and instances of that concept. This knowledge representation model supports the evolution of concepts as well as data. Adding a new concept to the knowledge structure involves correct placement of the node in the hierarchy by attaching links to higher level concepts and describing only how this concept differs from existing concepts. New data instances are attached as leaf nodes to the bottom of their concept hierarchy. Semantic networks play a role in many expert system shells (e.g., KL-ONE, described in Brachman [1978], ART [Clayton 1985], NIKL [Moser 1983]).

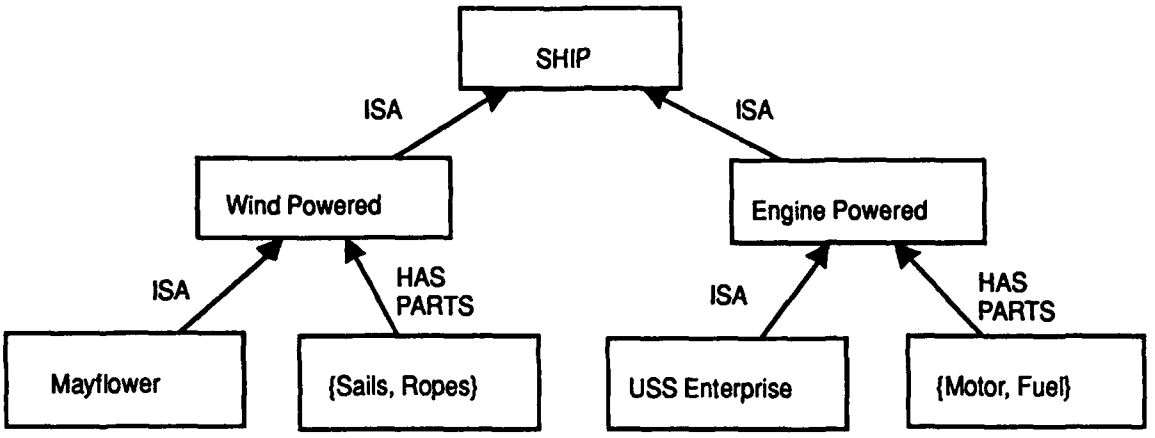


Figure A.3—Ship semantic net

A.1.3 Frame Based

A frame is a network of nodes and relations organized in a hierarchy much like a semantic net. The top node in a frame is usually organized around a general concept. The lower nodes and links represent more specific instances, concepts, and properties of the general concept. The lower nodes contain a collection of attribute value pairs, *slots*. In addition to values, slots can have attached procedure properties that take arbitrary actions when the value of a slot changes. Typical procedures in this class include if-added, if-removed, and if-needed. An if-added procedure is executed when new information is placed in the slot. An if-removed procedure is executed when information is deleted from a slot. An if-needed procedure is executed when information is needed from the slot, but the slot is empty. The primary difference between semantic networks and frames is the procedures attached to the slots in the nodes. The frame mechanism can be used to represent generic objects such as a restaurant. It can also support the customization of that restaurant to be a specific instance, e.g., an Italian restaurant. In addition, the restaurant frame can contain information about behavior patterns that occur in restaurants, ordering a meal, eating, paying the bill. It is easy to see that with sufficient refinement very complex restaurant behavior can be modeled. KEE [Kunz 1984] and SRL [Wright 1983] are expert system tools that make extensive use of frame based technology. SRL supports the construction of sophisticated hierarchical frame inheritance.

A.1.4 Object Oriented

Object oriented expert systems consist of objects, called actors, capable of exhibiting behavior by responding to messages and exhibiting behavior which results in sending messages to other objects. The objects of an object oriented expert system are similar to the frames of a frame based expert system. The uniqueness of object oriented expert systems lies in its effective use of message passing. Objects send and receive messages. An object upon receiving a message consults its local rules and database to decide the correct action to perform. Object oriented expert systems are used in a wide variety of applications. Programming languages, such as SMALLTALK [Goldberg 1983] and LOOPS [Bobrow 1983], have been developed to support the object oriented paradigm. There are object oriented simulation languages such as ROSS [McArthur 1984] and ERIC [Hilton 1987] and there is at least one commercially available object oriented DBMS, GEMSTONE [Maier 1986].

A.1.5 Procedure Oriented

Procedure oriented expert systems encode intelligent behavior in subroutines. This method of packaging code offers performance advantages by reducing unnecessary

duplication. A Knowledge Engineer develops the expert system's behavior by nesting procedures. In this way a high level "expert language" can be composed by appropriate subroutine nesting. The disadvantage of this method lies in its flexibility and explainability. Procedure oriented expert systems and systems designed using top-down structured methodology work well on their initially designed tasks. However, as the computer system matures its structure becomes a patch-work of changes to support feature enhancements. This makes each new change more difficult because of the depth of system understanding that the system analyst must possess. Explanation components of procedure oriented expert systems are difficult to construct because of the recursive nature of the nested procedure calls. Adding the extra record keeping structure and explanation generation component is almost equivalent to constructing a rule based expert system shell. The most common languages for procedure oriented expert system construction are the various LISP dialects and C.

A.1.6 Logic Based

Logic based expert systems use predicate calculus to construct the knowledge base and organize its execution. Logic based expert systems encode knowledge as a set of declarative clauses in predicate logic. Each clause is of the form :

```
consequent :- antecedent-1, antecedent-2, ... antecedent-n
```

where each of the antecedents are predicates that can be tested for truth value and the consequent is a predicate which is true if its antecedents can be proved true. The style of reasoning in logic based expert system is much like theorem proving. Given a query assertion, a *goal*, the knowledge base of consequents are pattern matched. If a matching consequent is found then the system attempts to prove the appropriately instantiated list of antecedents as *subgoals*. When all of the subgoals are proved true then the initial goal is proved. This style of knowledge encoding is very similar to the rule based technique of backward chaining; however, the search strategy is usually more rigidly controlled. Languages such as PROLOG support this style of programming.

A.1.7 Induction

The development of expert system rules from existing databases of information is a challenging process. To aid in this process some expert system shell vendors provide tools to develop rules by induction from an existing base of facts. This process offers a convenient way to obtain an initial set of rules in the development of an expert system. The expert system shell RULEMASTER [Michie 1984] provides a rule induction tool.

A.1.8 Access Oriented

Access oriented expert systems encode intelligent behavior by attaching procedures to data. These procedures may be called *demons* or *triggers*. Demon procedures are activated when the data are created, changed, or read. Demons can also be used to monitor external devices. Access oriented expert systems facilitate the use of real time sophisticated graphical displays. The term *active value* is also used to describe the concept of visually monitoring important system variables. Examples are the LOOPS [Bobrow 1983] access oriented programming language which supports a collection of dials and gauges and STEAMER [Hollan 1984], an expert system for tutoring students in the operation of a steam propulsion plant, which also utilizes dials and gauges.

A.1.9 Integrated

Integrated expert systems combine several different methods to improve expressiveness and completeness. The degree to which the various methods are integrated in any one tool varies widely. An area of expert system technology that currently has received little attention is the integration/embedding of expert systems into traditional data processing. Adding expert system components to existing databases and software systems is an

important next step in the evolution of expert systems from the laboratory to industry. KEEconnection™ [Parker 1987] and NEXPERT-OBJECT™ [Neuron Data 1987] are examples of the beginnings of this integration process. Expert systems such as ART and KEE contain most of the available expert system paradigms.

A.2 EXPERT SYSTEM CHARACTERISTICS

A.2.1 Arithmetic and Logical Processing

Expert system languages usually provide access to arithmetic and logical operators. The set of operators offered is usually the set of operators supported in the implementation language. For instance, an expert system tool implemented in LISP offers basic LISP arithmetic and logical operators. However, the rich set of data types in the underlying language and/or supported by the hardware processor may not be available in the expert system shell. In particular, extended precision arithmetic, infinite length strings, functional objects, and other hardware specific facilities may not be supported. Expert system shells that are available on multiple types of hardware usually support only the hardware capabilities of the minimum configuration.

A.2.2 Certainty Handling

The support of reasoning under uncertainty can be important in some expert systems development. In some cases the uncertainty can be managed by the Knowledge Engineer explicitly without expert system shell support. However, if the degree of uncertainty must be propagated through the system during the reasoning process then expert system support is essential. There are many types of uncertain data and several different methods of representing and combining uncertain information, such as Bayesian logic, certainty factors, fuzzy logic, and Dempster-Shafer. Each technique is effective at representing knowledge that fits its model of uncertainty. However, no single technique spans all uncertainty classes equally well.

The **Bayesian logic** method is best suited to knowledge that is a culmination of information. Prospector uses the Bayesian method to represent and combine uncertain evidence of mineral deposits. **Certainty factors** are numerical representations of the degree of belief in an assertion. MYCIN uses certainty factors to indicate the strength of suggestive evidence for a specific organism based on tests and symptoms. The **fuzzy logic** method uses concepts from fuzzy set theory to represent uncertainty. Fuzzy set membership is based on a probabilistic distribution. The **Dempster-Shafer** method uses intervals to represent ranges of belief. Degrees of belief that are present in common speech such as likely, unlikely, unknown, true, and false can be represented as Dempster-Shafer intervals on the range [0,1] by [0.75,1], [0,0.25], [0,1], [1,1], and [0,0], respectively.

A.2.3 Concurrency

The development of larger expert systems involves the consideration of using concurrency. The use of parallel processing architectures in an expert system can be important in developing high speed intelligent systems in areas like imagery, signal understanding, and continuous modeling. In order to manage large scale problems it is sometimes necessary to use multiple distributed computers. Expert systems that are to be integrated into existing distributed systems, i.e., large financial networks, distributed database systems, and computer integrated manufacturing, must support expert system development in a distributed computing environment. The additional hardware and software complexities of concurrent processing should be an integral part of the expert system shell to facilitate rapid system development.

A.2.4 Consistency Checking

The correctness of information in an expert system is central to its success. Expert system tools offer various syntactic and semantic components to improve system quality. The areas of consistency checking supported include rule bases and knowledge bases. In some expert system tools a syntax directed editor is provided to facilitate the development of "correct" rules. These tools support the rapid development of expert systems by locating many of the obvious bugs. However, they should not be relied upon as a guarantee of correctness; it is still possible to misspell an object name exactly like some other object name in the system and have it pass unnoticed into the rule set. Few if any expert systems provide consistency checking across rules.

A.2.5 Documentation Development

The development of on-line and off-line expert system documentation is an important component of the complete fielded systems. Under ideal conditions this documentation is developed at the same time as the system. Many tools offer skeleton documentation structures in the rules and knowledge bases to facilitate annotation by knowledge engineers during development. Most expert systems are developed through a process of successive refinement. It is important to record the evolving assumptions and rationale of the expert system as the development proceeds.

A.2.6 Explanation

The ability of an expert system to explain its actions separates it from traditional programming. Explanations in an expert system can occur at many levels. The simplest level is execution tracing, a running transcript of rule executions and knowledge base changes. The next level is the capability of explaining questions that it asked of the user. These explanations are usually a backward trace of the current rule nesting and supporting knowledge base facts that demonstrate the importance of knowing the answer to the posed question. The depth of the current rule nesting and the complexity of individual rules can make this "complete backward trace" style of explanation difficult to follow. Hence, the next level of explanations involves the generation of intelligent explanations from the current rule nest and knowledge base structure. Expert system shells usually provide hooks for Knowledge Engineer generated rules/code to manage this explanation task. As an expert system grows in complexity the difficult task of generating good explanations also gets more difficult. Not only is the expert system complex, but the Knowledge Engineer must also consider that generated explanations address users with various degrees of sophistication and expert system understanding.

A.2.7 Fielded System

Some expert system shells provide separate support for preparing and deploying an expert system. The rich development environment that facilitates the building of the expert system is different from the environment in the fielded system. The degree to which the end-user can change the expert system rules and the information in the knowledge base of the fielded system is usually limited. In addition, direct access to the underlying expert system shell's capabilities is limited to those available in the **runtime** version. Most expert system shell companies offer a **runtime** version of their product with reduced capabilities at a reduced cost. The expert system utilities that support the process of deploying a completed expert system on various delivery machines usually include machine specific compilers that produce fielded expert systems modules that contain only the requisite development components. Thus fielded expert systems require less sophisticated hardware systems and run faster than their development counterparts.

A.2.8 Inference Control

The development of an expert system involves building a knowledge base of information and collecting domain heuristics in rules. The true power of an expert system is in effective management of the reasoning process by utilizing the appropriate knowledge base and rule base components. Several paradigms for controlling the inferencing process are available and each has its advantages and disadvantages. Most expert system shells provide only one or just a few of the paradigms. Only in the largest and most expensive expert system shells will one find an integrated set of all the current paradigms.

Forward chaining and backward chaining through the rules have previously been discussed as inferencing paradigms. In each case **conflict resolution**, deciding which of the rules currently ready to fire should be next to execute, plays an important part. Some systems offer user hooks to augment the rule scheduling process. Others provide the Knowledge Engineer with syntactic tools for controlling rule execution order, e.g., shortest rule next, first rule entered next, and rule priority.

The expressiveness of the rule language is directly related to the power of the expert system. The rule language is a combination of patterns and operators. The pattern matching capabilities of the rule matching system and the complexity of pattern expressions enhance the rule expressiveness. The rule operators, such as iteration, conjunction, negation, disjunction, and logical dependency, support the construction of complex rules. As rule languages grow in complexity the old technique of estimating the size of an expert system by rule count becomes less and less valid.

The style of interactive expert system reasoning has led to the development and support of several new reasoning paradigms. These paradigms were developed to support sound reasoning during the expert system development process without requiring the Knowledge Engineer to propagate all knowledge base and rule base changes. **Inheritance methods**, the first paradigm, propagate knowledge base changes within inheritance hierarchies in an attempt to maintain knowledge base consistency without requiring the user to update all affected fields. The second paradigm, **truth maintenance**, is a generalization of inheritance. Inheritance was not powerful enough by itself to support the logical consistency of the knowledge base. Truth maintenance systems include sophisticated inheritance components and rule base components to attempt the difficult task of logical consistency.

The task of maintaining logical consistency can be exceedingly difficult. In order to illustrate the issues involved, we shall look at an example. A fact that was entered into the knowledge base in the past is later denied. The truth maintenance system must remove the fact and **ALL** side effects. The side effects can include information inherited from the fact and rules that fired because the fact was true. These inherited facts and fired rules potentially caused other changes that must be undone. But how does one undo destructive changes, e.g., deleted database records or printed output resulting from earlier facts (later changed)? Undoing the fact propagation can be very time consuming and it is difficult to prove that complete truth maintenance has occurred without severe rule language restrictions.

The next paradigm, **hypothetical reasoning**, is a different approach to the truth maintenance issue. It allows the user to specify the tentative nature of a hypothesis. This style of reasoning, often called what-if reasoning, allows the Knowledge Engineer to establish fixed points in the knowledge base to reason from and to mark information that is added to the knowledge base as potentially temporary. If the what-if path turns out to be correct, some expert system shells offer a **believe** operator to permanently incorporate the knowledge base changes.

A.2.9 Integration

The development of an expert system often arises from the need to apply heuristic knowledge to a problem that has not been satisfactorily solvable through algorithmic methods. The obvious approach is to integrate the algorithmic components into a heuristic

framework. The support for integrating expert systems with other software and databases is just evolving in the expert system tools world. Most expert system shells assume that the Knowledge Engineer is developing a stand-alone system completely contained within the expert system development environment. The integration process is not straight forward for many of the tools because of the advanced internal tool architectures required to support heuristic reasoning. In particular, knowledge bases and rule bases are not stored in traditional database architectures. Hence, an existing traditional database may seem the ideal knowledge base for an expert system during the design process before beginning the prototyping or implementation, but may require significant rework for expert system integration. To facilitate the integration process some expert system shells support call-in/call-out to other languages and special interfaces to popular database systems.

A.2.10 Internal Access

To facilitate support of special databases and software systems, Knowledge Engineers often resort to utilizing knowledge about the internal structure and source code of an expert system shell. This process is **risky**. Expert system tool vendors will rarely guarantee that they will not change the internal architecture or function names throughout a tool's lifetime. It is always best to work with the tool vender to extend the tool's capabilities. These extensions will often become part of future vendor-released versions of the tool, thereby relieving the user from long term support issues.

A.2.11 Knowledge Acquisition

The building of the knowledge base and rules of an expert system can be a formidable task. Some tools offer assistance in building models and deriving rules from existing information. These tools are helpful in the design process, but they need guidance and validation from the user. They can also help in the analysis of an existing system by noticing rule clusterings during execution which tend to generalize the understanding of the domain model.

A.2.12 Knowledge-Base Editing

During the development process both the rule base and the knowledge base are edited by the Knowledge Engineer. Some expert system shells provide special tools to facilitate this editing process. These tools support editor-like capabilities enhanced by such features as graphical displays of rule networks, syntax directed editors, and limited forms of knowledge base consistency maintenance. These special purpose extensions are a valuable asset to the original Knowledge Engineer as well as those tasked with long term maintenance of the expert system because they raise the level of the man machine dialogue.

A.2.13 Meta-knowledge

In developing an expert system it is often important to represent knowledge about knowledge. This higher level description is used to focus the reasoning process. Meta knowledge about knowledge bases or rule sets can reduce the reasoning time required. This meta knowledge is represented in rules that control the inferencing process and in the knowledge base structures. Meta knowledge also plays an important part in distributed expert systems by capturing the distributed reasoning architecture and using this knowledge to effectively distribute the reasoning process. A **blackboard** architecture is often used in distributed expert systems. The blackboard serves as a centralized location for the distributed expert system components to share information. When an expert system discovers information that may be useful to other expert systems in the distributed environment it posts the information on the blackboard for all to see. Hearsay III used a blackboard architecture to share information among various expert systems in an attempt to understand speech in a limited domain.

A.2.14 Optimization

The performance of an expert system is usually a feature that is addressed at the end of development. During the initial stages it is usually sufficient to get the right answer. After the expert system generates acceptable responses, then the matter of improving the performance to meet acceptable interaction standards is addressed. Some expert system shells offer features such as rule compilation, caching, and intelligent look-ahead to facilitate optimization. In addition, re-examining the effectiveness of the overall reasoning approach on representative test data can lead to insight in the process of rule restructuring. An example of this optimization process is replacing a rule that handles the general case with several rules such that the most common occurrence is handled quickly and the rarer cases follow the general rule.

A.2.15 Presentation

Design of the user interface of an expert system is important to system acceptance. Issues such as ease of data entry, use of color in displays, graphical capabilities, multiple windows, and mouse-like pointing devices are raised when discussing user friendly interfaces. The perceived friendliness of the user interface is often listed as the determining factor in expert system tool choice. The initial reaction of a user to a new expert system will usually be affected by the friendliness of the user interface. Some expert system shells provide the Knowledge Engineer with a set of interface tools to manage the appearance issues such as windows, mouse input, animation, graphics, spreadsheets, tables, and forms. Using expert system shells with these capabilities built in allows the Knowledge Engineer to forgo the task of building and integrating a reasonable interface system. In addition, those vendors that support multiple hardware platforms have often generalized the interface language supported by their shell to the point where the Knowledge Engineer's interface will work on all support architectures.

A.2.16 Real time

Expert systems that must perform in a real time environment require special design considerations. Typical expert systems manage large knowledge bases and use reasoning techniques that involve the construction of many partial solutions. Both of these processes generate computational garbage that must be recycled. The recycling process, garbage collecting, changes the responsiveness of the expert system, even in the parallel garbage collector case. Large expert systems usually require large address spaces, hence they rely on paging systems which add time variability. Real time usually means short elapse time between the inputs, perhaps from sensors, and the outputs, i.e., process control. In the area of computer integrated manufacturing, process time on the order of seconds is a common requirement. To achieve this performance, special purpose hardware such as parallel computer and distributed systems are often used.

A.3 GLOSSARY OF EXPERT SYSTEM TERMS

Access-oriented methods	Programming methods based on the use of probes that trigger new computations when data are changed or read.
Active value	A procedure invoked when program data are changed or read, often used to drive graphical displays of gauges that show the values of the program variables.
AI	Artificial intelligence

AIS	Automated information system
Algorithm	A formal procedure guaranteed to produce correct or optimal solutions.
Artificial intelligence	The part of computer science concerned with developing intelligent computer programs.
Attribute	Describes a property of an object.
Backtracking	Returning back to a known step during a search procedure. If a choice has led to an unacceptable result, the program returns to the previous specified point and opts for another choice.
Backward chaining	An inference method where the system starts with what it wants to prove, e.g., Z, and tries to establish the facts it needs to prove Z.
Blackboard architecture	A way of representing and controlling knowledge based on using independent groups of rules called knowledge sources that communicate through a central database called a blackboard.
Breadth-first search	A search technique that evaluates every item at a given level of the search space before proceeding to the next level.
Break package	A mechanism in a programming or knowledge engineering language for telling the program where to stop so the programmer can examine the values of variables at that point.
CAI	Computer-aided instruction.
Certainty factor	A number that measures the certainty or confidence one has that a fact or rule is valid.
CF	Certainty factor.
Commercial system	The system is a production model being used on a regular commercial basis.
Control	Expert system application: governing overall system behavior.
Cooperating knowledge sources	Specialized modules in an expert system that independently analyze the data and communicate via a central, structured database called a blackboard.
CS	Computer science.

Database	The set of facts, assertions, and conclusions used to match against the rules in a rule-based system (often has a broader meaning).
DB	Database.
DBMS	Database management system, generally a commercially available product that manages "facts."
Debugging	Expert system application: prescribing remedies for malfunctions.
Decision support system	A passive software tool used to model/forecast various problems to assist in the decisionmaking process, typically using matrix-type data structures.
Declarative statement	A fact; an assertion that is true.
Deduction	The deriving of a conclusion by reasoning. An inference process in which the conclusion follows necessarily from the premises.
Demon (daemon)	A procedure activated by the changing or accessing of values in a database.
Demonstration prototype	The system solves a portion of the problem undertaken, suggesting that the approach is viable and system development is achievable.
Depth-first search	A search technique that evaluates only one item at a given level of the search space before proceeding to the next level.
Design	Expert system application: configuring objects under constraints.
Diagnosis	Expert system application: inferring system malfunctions from observables.
Domain	The universe of data and knowledge contained in the expert system, usually related to a common set of problems or tasks.
Domain expert	An individual, who by training, education, experience, or insight, is able to solve a particular type of problem that most other people cannot solve nearly as efficiently or effectively.
Domain knowledge	Knowledge about the problem domain; e.g., knowledge about geology in an expert system for finding mineral deposits.
DSS	Decision support system.

End-user	The person who uses the finished expert system; the person for whom the system was developed.
ES	Expert system.
Example based	An expert system structure whereby the knowledge base is entered into the system through examples. The inference engine then deciphers these examples by means of induction to form the rule base.
Exhaustive search	A problem-solving technique in which the problem solver systematically tries all possible solutions in some brute-force manner until it finds an acceptable one.
Expert system	A computer program using expert knowledge to attain high levels of performance in a narrow problem area.
Expert system building tool	The programming language and support package used to build the expert system.
Explanation facility	That part of an expert system that explains how solutions were reached and justifies the steps used to reach them.
FD	Functional description.
Field prototype	The system displays good performance with adequate reliability and has been revised based on extensive testing in the user environment.
Forward chaining	An inference method where rules are matched against facts to establish new facts.
Frame	A knowledge representation method that associates features with nodes representing concepts or objects. The features are described in terms of attributes (called slots) and their values.
Frame-based methods	Programming methods using frame hierarchies for inheritance and procedural attachment.
Fuzzy logic	When no clear concise boundary exists between situations where a concept applies and situations where it does not apply. Fuzzy logic is a gradual transitioning where a network of assertions control the success or failure of subsequent logic.
Garbage collection	A set of methods for recovering areas of address space that are no longer being used. Once the memory is recovered, it can be reused to represent new objects. Garbage collection can be done parallel to program processing or can interrupt program processing.

General-purpose KE language	A computer language designed for building expert systems and incorporating features that make it applicable to different problem areas and types.
Generate and test	A problem-solving technique involving a generator that produces possible solutions and an evaluator that tests the acceptability of those solutions.
Granularity	The level of detail in a chunk of information.
Heuristic	A rule of thumb or simplification that limits the search for solutions in domains that are difficult and poorly understood (also heuristic rule).
Heuristic search	A procedure by which a computer attacks problems by a trial-and-error method and which may involve the act of learning.
Induction	The act, process, result or an instance of reasoning from part to a whole, from particulars to generals, or from the individual to the universal.
Inference	An implied relationship of one object to another, allowing new facts to be derived from existing facts.
Inference chain	The sequence of steps or rule applications used by a rule-based system to reach a conclusion.
Inference engine	That part of a knowledge-based system or expert system that contains the general problem-solving knowledge.
Inference method	The technique used by the inference engine to access and apply the domain knowledge, e.g., forward chaining and backward chaining.
Inference net	All possible inference chains that can be generated from the rules in a rule-based system.
Inheritance (inheritance hierarchy)	A feature of semantic network data structures: the ability of one node to inherit characteristics of other nodes that are related to it. A structure in a semantic net or frame system that permits items lower in the net to inherit properties from items higher up in the net.
Instruction	Expert system application: diagnosing, debugging, and repairing student behavior.
Interface	An area of communication of knowledge between two or more parties or systems.
Interpretation	Expert system application: inferring situation descriptions from sensor data.

Interpreter	The part of the inference engine that decides how to apply the domain knowledge.
I/O	Input/output.
IPR	In-process review.
KB	Knowledge base.
KBA	Knowledge Base Administrator.
KBS	Knowledge-based system.
KE	Knowledge Engineer.
Knowledge acquisition	The process of identifying, documenting, and analyzing the information processing behavior of the expert.
Knowledge acquisition techniques	See: on-site observation; problem discussion; problem description; problem analysis; system refinement; system examination; system validation.
Knowledge base	The portion of a knowledge-based or expert system that contains the domain knowledge.
Knowledge Engineer	The person who designs and builds a knowledge-based system.
Knowledge engineering	The process of building knowledge-based systems.
Knowledge representation paradigm	A knowledge representation method (see: frame, rule, semantic net).
Knowledge representation	The process of structuring knowledge about a problem in a way that makes the problem easier to solve. A structure (format) in which knowledge is stored that allows the system to understand the relationships of the data in the knowledge base.
Knowledge	The information a computer program must have to behave intelligently.
Knowledge-based system	A program in which the domain knowledge is explicit and separate from the program's other knowledge.
KR	Knowledge representation.
Links (arcs)	Relates objects (nodes) and descriptors in semantic network data structures.
List structure	A collection of items enclosed by parentheses where each item can be either a symbol or another list, e.g., (BALL SHOE (Y5 BILL) 23 (CAT 7)).

Logic	A technique for drawing inferences that relies on formal rules for manipulating symbols.
Logic-based methods	Programming methods that use predicate calculus to structure the program and guide execution.
Meta-knowledge	Knowledge about knowledge. Knowledge about the use and control of domain knowledge in an expert system. Knowledge in an expert system about how the system operates or reasons.
Meta-rule	A rule that describes how other rules should be used or modified.
Monitoring	Expert system application: comparing observations to expected outcomes.
Monotonic reasoning	Where all values concluded for an attribute remain true for the duration of the program run.
Multiple lines of reasoning	A problem-solving technique in which limited numbers of possibly independent approaches to solving the problem are developed in parallel.
Natural language	The standard method of exchanging information between people, such as English (to be contrasted with artificial languages, such as programming languages).
Nonmonotonic reasoning	Any facts that once defined (true or false) may be retracted.
Object-oriented methods	Programming methods based on the use of items called objects that communicate with one another via messages.
On-site observation	Watch the expert solving real problems on the job (knowledge acquisition technique).
OR	Operations research.
Paradigm	A pattern, model, or example.
Parallel processing	A computer processing technique that allows the computer to perform multiple processes at the same time, in parallel.
Parser	A translation program that converts an input file into an output file which can then be interpreted by a program.
Pattern matching	An AI technique that recognizes similarities between patterns and objects or events.
Planning	Expert system application: designing actions.

Plausible reasoning	A guess that is made when attempting to cope with incomplete knowledge, when solutions are plentiful and any path will do, or when trying to limit the scope of the domain and converge upon the solution rapidly.
PM	Project manager.
PN	Progress Notebook.
Predicate calculus	A formal language of classical logic that uses functions and predicates to describe relations between individual entities.
Prediction	Expert system application: inferring likely consequences of given situations.
Premise	The "if" part of a production rule. If the premise of a rule being tested is true, the action ("then") part of the rule is evaluated.
Probability propagation	The adjusting of probabilities at the nodes in an inference net to account for the effect of new information about the probability at a particular node.
Problem analysis	Present the expert with a series of realistic problems to solve aloud, probing for the rationale behind the reasoning steps (knowledge acquisition technique).
Problem description	Have the expert describe a prototypical problem for each category of answer in the domain (knowledge acquisition technique).
Problem discussion	Explore the kinds of data, knowledge, and procedures needed to solve specific problems (knowledge acquisition technique).
Problem reformulation	Converting a problem stated in some arbitrary way to a form that lends itself to a fast, efficient solution.
Problem solving	The process of starting in an initial state and searching through a problem space in order to identify the sequence of operations or actions that will lead to a desired goal. Weak problem-solving techniques are domain-independent. Strong problem-solving techniques are domain-dependent and exploit knowledge to achieve greater performance.
Problem-oriented language	A computer language designed for a particular class of problems, e.g., FORTRAN designed for efficiently performing algebraic computations, COBOL with features for business record keeping.
Problem space	A method of structuring the problem into a two-dimensional matrix composed of problem categories by reasoning subcomponents.

Procedure-oriented methods	Programming methods using nested subroutines to organize and control program execution.
Production model	The system exhibits high quality, reliable, fast, and efficient performance in the user environment (also called fielded system).
Production rule	A condition/action rule that produces changes that result in a new state for another condition/action rule to be applied, and so on. The most common format of this data structure is IF-THEN.
Programming language	An artificial language developed to control and direct the operation of a computer.
Protocol	A record of the mental and/or physical actions taken when accomplishing a task. Protocols need to be analyzed to determine relevant knowledge and reasoning strategies.
Pruning	Reducing or narrowing the alternatives, normally used in the context of reducing possibilities in a branch tree structure.
QA	Quality assurance.
Real-world problem	A complex, practical problem whose solution is useful in some cost-effective way.
Recursive	Nested iterations, a procedure that is able to call itself to solve a subprocess.
Repair	Expert system application: executing plans to administer prescribed remedies.
Representation	The process of formulating or viewing a problem so it will be easy to solve.
Research prototype	The system displays credible performance on the entire problem but may be fragile due to incomplete testing and revision.
Robustness	That quality of a problem solver that permits a gradual degradation in performance when it is pushed to the limits of its scope of expertise or is given faulty, inconsistent, or incomplete data or rules.
ROI	Return on investment.
Rule	A formal way of specifying a recommendation, directive, or strategy, expressed as IF premise THEN conclusions or IF condition THEN action.

Ruleset	A collection of rules that constitutes a module of heuristic knowledge.
Rule-based methods	Programming methods using IF-THEN rules to perform forward or backward chaining.
Scaling problem	The difficulty associated with trying to apply problem-solving techniques developed for a simplified version of a problem to the actual problem itself.
Scheduler	The part of the inference engine that decides when and in what order to apply different pieces of domain knowledge.
Schema	A frame-like representation formalism in a knowledge engineering language (e.g., SRL).
Search	The process of skillfully looking through the set of possible solutions to a problem in order to efficiently find an acceptable solution.
Search space	The set of all possible solutions to a problem.
Semantic net (semantic network)	A knowledge representation method consisting of a network of nodes standing for concepts or objects connected by arcs describing the relations between the nodes.
Sequential processing	The traditional computer processing technique of performing actions one at a time, that is, in sequence.
Skeletal KE language	A computer language designed for building expert systems and derived by removing all domain-specific knowledge from an existing expert system.
Skill	The efficient and effective application of knowledge to produce solutions in some problem domain.
Slot	An attribute associated with a node in a frame system. The node may stand for an object, concept, or event; e.g., a node representing the object employee might have a slot for the attribute name and one for the attribute address. These slots would then be filled with the employee's actual name and address.
Support environment	Facilities associated with an expert system building tool that help the user interact with the expert system. These may include sophisticated debugging aids, friendly editing programs, and advanced graphic devices.
SW/HW	Software/hardware
Symbol	A string of characters that stands for some real-world concept.

Symbol-manipulation language	A computer language designed expressly for representing and manipulating complex concepts. Examples are LISP and PROLOG.
Symbolic reasoning	Problem solving based on the application of strategies and heuristics to manipulate symbols standing for problem concepts.
System examination	Have the expert examine and critique the prototype system's rules and control structure (knowledge acquisition technique).
System refinement	Have the expert give you a series of problems to solve using the rules acquired from the interviews (knowledge acquisition technique).
System validation	Present the cases solved by the expert and prototype system to other outside experts (knowledge acquisition technique).
T&E	Test and evaluation.
Tool	A shorthand notation for expert system building tool.
Tool builder	The person who designs and builds the expert system building tool.
Tools for knowledge engineering	Programming systems that simplify expert system development. They include languages, programs, and facilities that assist the Knowledge Engineer.
Toy problem	An artificial problem, such as a game or an unrealistic adaptation of a complex problem.
Tracing facility	A mechanism in a programming or knowledge engineering language that can display the rules or subroutines executed, including the values of variables used.
Units	A frame-like representation formalism employing slots with values and procedures attached to them.
User	A person who uses an expert system, such as an end-user, a domain expert, a Knowledge Engineer, a tool builder, or a clerical staff member.
WM	Working memory.
WRT	With respect to.

A.4 GLOSSARY OF LOGISTICS TERMS

ACO	Administrative Contracting Officer.
ADAM	Aerial Port Documentation and Management.
ADP	Automatic data processing.
ADPSO	Automatic Data Processing Support Office.
AFLC	Air Force Logistics Command.
ALC	Air Logistics Center.
ALMSA	Automated Logistics Management Systems Activity.
AMC	Army Materiel Command.
AMCL	Approved MILSTRIP (Military Standard Requisitioning and Issue Procedures) Change Letter.
AMRF	NBS Automated Manufacturing Research Facility.
ANSI	American National Standards Institute.
ARPANET	(Defense) Advanced Research Projects Agency Network.
AT&T	American Telephone and Telegraph.
ATCMD	Advance Transportation Control and Movement Data.
ATE	Automated test equipment.
AUTODIN	Automatic digital network.
AUTOVON	Automatic voice network.
Base-level stratification	Once requirement determination is made for an item for every purpose, the method of base-level stratification becomes necessary to combine all these data to develop a single requirement.
BIT	Built-in test.
bps	Bits per second.
CAD	Computer-aided design.
CALS	Computer-aided logistics support.
CAM	Computer-aided manufacturing.

CASREP	Casualty Report (Navy Casualty Summary Report).
Cataloging	Naming, identifying, classifying, and numbering items of property.
CCA	Computer Corporation of America.
CCSS	Commodity Command Standard System.
CDC	Control Data Corporation.
CDR	Contractor Deficiency Report.
CIM	Computer integrated manufacturing.
CINC	Commander-in-Chief.
COBOL	Common Business Oriented Language.
CODASYL	Conference on Data Systems Language.
Configuration management	Set of activities for effective management of systems configuration data; as parts change over time, timely configuration management change notices allow supply operations to identify parts for accessing at the earliest possible time.
Contract administration	Comprises all the actions the government must take with the contractor until the material or service has been delivered, accepted, and paid for, and the contract closed out.
CONUS	Continental United States.
CPU	Central processing unit.
DAAS	Defense Automatic Addressing System.
DAASO	Defense Automatic Addressing System Office.
DAISY	DRMS Automated Information System.
DAR	Destination Acceptance Report.
DBM	Database machine.
DBMS	Database management system.
DCA	Defense Communications Agency.
DCAS	Defense Contract Administration Service.
DCASR	Defense Contract Administration Service Region.

DCSC	Defense Construction Supp'y Center.
DDBMS	Distributed database management system.
DDN	Defense Data Network.
DEPRA	Defense European and Pacific Redistribution Activity.
DFARS	Defense Federal Acquisition Regulation Supplement.
DIAL	Data interchange at the application level.
DIDS	Defense Integrated Data System.
Distribution and redistribution	Logistics processes that cover positioning of materiel at specific storage points (distribution) and movement of materiel between storage points (redistribution). Distribution systems consist of a complex series of echelons of supply.
DLA	Defense Logistics Agency.
DLANET	Defense Logistics Agency Telecommunications Network.
DLSC	Defense Logistics Services Center.
DLSS	Defense Logistics Standard Systems.
DLSSO	Defense Logistics Standard Systems Office.
DMS	Data management system.
DoD	Department of Defense.
DoDAAD	Department of Defense Activity Address Directory.
DoDAAF	Department of Defense Activity Address File.
DoDD	Department of Defense Directive.
DoDI	Department of Defense Instruction.
DRMO	Defense Reutilization and Marketing Office.
DRMS	Defense Reutilization and Marketing Service.
DSS/ALOC	U.S. Army Direct Supply Support/Air Line of Communication.
DTS	Defense Transportation System.
EBDI	Electronic business data interchange.

Inventory control	Includes maintenance and replenishment of stock levels so that items are supplied consistent with need; overall investment in inventories is kept to a minimum, consistent with need; supply workload (including procurement actions and stock status and transaction reporting) is controlled in both detail and frequency.
IRIS	Interrogation Requirements Information System.
IRM	Information Resources Management.
ISDN	Integrated Services Data Network.
ISO	International Standards Organization.
JCS	Joint Chiefs of Staff.
JDA	Joint Deployment Agency.
JDS	Joint Deployment System.
JOPES	Joint Operations, Planning, and Execution System.
kbps	Kilobytes per second.
L&MM	Logistics and Materiel Management.
LAN	Local area network.
LASE	Logistics Assets Support Estimate.
LIDS	Logistics Information Data Service.
LIF	Logistics Intelligence File.
LOGDESMAP	Logistics Data Element Standardization and Management Program.
LOGDRMS	Logistics Data Resource Management System.
LOGNET	Logistics network.
LSAO	Logistics Systems Analysis Office.
M3S	Marine Corps Standard Supply System.
MAC	Military Airlift Command.
MAISRC	Major Automated Information System Review Council.
MAPAD	Military Assistance Program Address Directory.
MAPAF	Military Assistance Program Address File.

Inventory control	Includes maintenance and replenishment of stock levels so that items are supplied consistent with need; overall investment in inventories is kept to a minimum, consistent with need; supply workload (including procurement actions and stock status and transaction reporting) is controlled in both detail and frequency.
IRIS	Interrogation Requirements Information System.
IRM	Information Resources Management.
ISDN	Integrated Services Data Network.
ISO	International Standards Organization.
JCS	Joint Chiefs of Staff.
JDA	Joint Deployment Agency.
JDS	Joint Deployment System.
JOPES	Joint Operations, Planning, and Execution System.
kbps	Kilobytes per second.
L&MM	Logistics and Materiel Management.
LAN	Local area network.
LASE	Logistics Assets Support Estimate.
LIDS	Logistics Information Data Service.
LIF	Logistics Intelligence File.
LOGDESMAP	Logistics Data Element Standardization and Management Program.
LOGDRMS	Logistics Data Resource Management System.
LOGNET	Logistics network.
LSAO	Logistics Systems Analysis Office.
M3S	Marine Corps Standard Supply System.
MAC	Military Airlift Command.
MAISRC	Major Automated Information System Review Council.
MAPAD	Military Assistance Program Address Directory.
MAPAF	Military Assistance Program Address File.

MCDN	Marine Corps Data Network.
MCWI	Motor Carrier Waybill Interchange.
MILSBILLS	Military Standard Billing System.
MILSCAP	Military Standard Contract Administration Procedures.
MILSPETS	Military Standard Petroleum System.
MILSTAMP	Military Standard Transportation and Movement Procedures.
MILSTEP	Military Supply and Transportation Evaluation Procedures.
MILSTRAP	Military Standard Transaction Reporting and Accounting Procedures.
MILSTRIP	Military Standard Requisitioning and Issue Procedures.
MODELS	Modernization of Defense Logistics Standard Systems.
MOV	Materiel Obligation Validation.
MRAD	Materiel Receipt Acknowledgment Document.
MRASDRS	Materiel Receipt Acknowledgment and Supply Discrepancy Reporting System.
MRC	Materiel Release Confirmation.
MRO	Materiel Release Order.
MRP	Materiel Returns Program.
MRP	Manufacturing requirements planning.
MSC	Military Sealift Command.
MTMC	Military Traffic Management Command.
MTMR	Military Traffic Management Regulation.
NARF	Naval Air Rework Facility.
NASA	National Aeronautics and Space Administration.
NAVSUP	Naval Supply Systems Command.
NBS	National Bureau of Standards.

Negotiation	Contracting by means of competitive or other-than-competitive proposals and discussions. Any contract awarded without using sealed bidding procedures is negotiated.
NIMSC	Nonconsumable Item Material Support Code.
NMCS	Not mission capable supply.
Non-demand based items/levels	Specific line items to support special projects or programs.
NSC	Naval Supply Center.
NSN	National Stock Number.
OCONUS	Outside the Continental United States.
O-D	Origin-Destination.
ODASD	Office of the Deputy Assistant Secretary of Defense.
OJCS	Organization of the Joint Chiefs of Staff.
OMB	Office of Management and Budget.
OSD	Office of the Secretary of Defense.
OSI	Open systems interconnection.
PC	Personal computer.
PCO	Procurement Contracting Officer.
PICA	Primary Inventory Control Activity.
PIP	Physical Inventory Program
Pipeline performance	Performance from date of requisition to the date materiel is offered to the consignee's transportation officer or delivered to the parcel post addressee.
P&L	Production and Logistics.
PMCL	Proposed MILSTRIP Change Letter.
POM	Program Objective Memoranda.
PPBS	Planning, Programming, and Budgeting System.
Primary item	Weapon system or major equipment.

Priority Group 1 requisitions	Standard procedures are needed for special processing of these in backorder status.
Procurement	The process of obtaining personnel, services, and equipment from the private sector through purchase order, contract, or other obligating documents.
PWA	Printed wiring assembly.
QDR	Quality Deficiency Report.
R&D	Research and development.
R&M	Reutilization and marketing.
RDBMS	Relational database management system.
RDD	Required delivery date.
RDF	Revised delivery forecast.
RFP	Request for proposal.
ROD	Report of Discrepancy.
RWI	Rail Waybill Interchange.
S/A	Service/agency.
SAMMS	Standard Automated Materiel Management System.
SC&D	Stock control and distribution.
Secondary item	Part(s) of a primary item (e.g., weapon system or equipment).
SICA	Secondary inventory control activity.
SIWSM	Secondary Item Weapons System Management.
SMP	Small mechanical parts.
SMPG	Supply Management Policy Group.
Solicitation	Act of advertising, etc., for goods, services, and facilities of requisite quality and within time needed at lowest reasonable cost using competitive bidding as much as possible.
SOW	Statement of work.
SPAR	Stock point ADP replacement.

SPLICE	Stock point logistics integrated communications environment.
SPN	Shipment Performance Notice.
SPR	Special program requirement.
SQL	Structured Query Language.
Supply	Supply should provide end items, equipment, and repair and replacement parts to operational and support units on an as-requested basis. Supply includes retail operations, wholesale inventory management, technical data management, and wholesale storage as major subfunctions for the MODELS analysis.
TCMD	Transportation control and movement data.
TCN	Transportation Control Number.
TCO	Termination Contracting Officer.
TDR	Transportation Discrepancy Report.
Technical data acquisition	Activities relating policies and procedures used to obtain needed technical data for maintenance and repair, spare parts procurement, and inventory management. For major end items the technical data are usually acquired from the R&D contractor as one product in the total set of contractual deliverables.
TIF	Test and Integration Facility.
TO	Technical Order.
TOA	Transportation Operating Agencies.
UCS	Uniform Communications Standard (for retail food industry).
UMMIPS	Uniform Materiel Movement and Issue Priority System.
WBS	Work breakdown structure.
WINS	Warehouse Information Network Standards.
WWMCCS	Worldwide Military Command and Control System.

Appendix B

DOCUMENT DESCRIPTIONS

Short descriptions of required/suggested documents can be found in this appendix. All descriptive material except for Sections B.6, B.8, and B.13 was taken from Draft DoD Standard 7935.1, "Automated Data Systems (ADS) Documentation Standards," 15 January 1988.

As noted in DoD Standard 7935.1, all of the document types may not be needed on any particular project. The project manager must determine early in the development process which of the document types will be needed for the project. DoD Standard 7935.1 also states that individual services or agencies may establish their own documentation specifications or document types in addition to those contained in the standard when the purpose of the new types does not duplicate the purpose of the specifications and types contained in DoD Standard 7935.1. The information contained in the additional specifications may range from technical information about specific hardware or software usage to management information about the development of the project. This Guide has introduced three new types of documents: Product Configuration Control Techniques, the Progress Notebook, and the Users Manual.

B.1 COMPUTER OPERATION MANUAL (OM)

The OM contains precise and detailed information on the control requirements and operating procedures necessary to successfully initiate, run, and terminate the expert system (ES). The OM is directed toward supervisory and operator personnel who are responsible for the efficient performance of the ES. All recovery and backup procedures should be specified. Since the ES can fail at several levels (i.e., ES, ES tool, workstation, AIS, and I/O device interfaces), the OM should describe how to ascertain, through the use of diagnostics and tests, the level at which a problem or failure is occurring. The readers of the OM are primarily interested in detailed information on the external characteristics and operating procedures of a computer program. In general, the OM should be written in a step-by-step fashion as opposed to an expository style in order to clarify and emphasize the procedures associated with the ES operation.

B.2 DATABASE SPECIFICATION (DS)

The DS is a technical document that describes in detail the data and knowledge used by the system, its source, acquisition techniques, system representation, where and how it is used, error handling techniques, removal/archiving methods, assumptions about the data, and implications of the data structure. All the information collected about source data should be documented in the DS. To achieve consistency in documentation, the following practices shall apply:

- Each graphic representation shall be followed by a narrative explanation.
- Each item of information shown in graphic representation shall be consistent with standardized data element names as shown in data element libraries.

B.3 END USER MANUAL (EM)

The EM describes how to use the ES in detail from an end-user perspective, including description of the workstation, how it operates, and its integration with AISs or special I/O devices. It should include a step-by-step tutorial that, if possible, can be run by the end-user on the ES workstation in demonstration or educational mode. The best way to organize the EM is dependent on the application and the end-users' view of the application. If the end-users conceptualize the system better from an application-category view than from a functional view, then the category organization should take precedence over the functional view, or the EM might cover each in a different section. Sections that should be included are: help, browsing through the knowledge base, error messages, automatic recovery techniques, and the means by which the user can comment on or express his/her dissatisfaction with the system (problem with knowledge or reasoning, awkward interface, poor browsing, etc.). Error handling must include all error messages the end-user could receive from the ES tool, operating system, and network as well as from the ES. At the least the end-user should be told where the error message came from and when it is necessary to tell a support person about the error.

B.4 FUNCTIONAL DESCRIPTION (FD)

The FD reflects the definition of the system requirements and provides the users with a clear statement of the system's operational capability. The FD is a tool for use by both computer and noncomputer-oriented personnel and should be written as much as possible in nontechnical language.

B.5 IMPLEMENTATION PROCEDURES (IP)

The IP is a tool for directing ES installation at locations other than the test site after ES testing has been completed. It may also be used to direct the implementation of major modifications or enhancements of an ES that has already been installed. Those parts of the document directed toward the staff personnel shall be presented in nontechnical language and those parts of the document directed toward the operations personnel shall be presented in suitable terminology.

B.6 PRODUCT CONFIGURATION CONTROL TECHNIQUES

This document describes the installed configuration control system that should be used to maintain the ES. It will be used by the Knowledge Base Administrator and by the ADP and communications personnel who maintain the interfaces. All configuration management information accumulated since the start of the Testbed Prototype effort should be included in this document.

B.7 MAINTENANCE MANUAL (MM)

The MM presents general information and specific technical information about the ES program for the Knowledge Base Administrator, the ES tool smith, and the ADP and/or network personnel who will maintain the interfaces.

B.8 PROGRESS NOTEBOOK (PN)

The purpose of the Progress Notebook is to document the history of all relevant project activities, some of which might otherwise never be permanently documented.

In principle, the PN is a living document, an historic trace of all alternatives and decisions, open issues, blind alleys pursued, and the like. It fills in the gaps left by the standard documents and serves as a navigational aid for reading through the other project documents. The PN contains a wide variety of information with varying degrees of sensitivity and serves as a central source of information for the project team as well as for future maintainers of the resulting ES.

For ease of reference and to avoid duplication, the PN should include the following standard documents:

FD	Functional Description
SS	System/Subsystem Specification
US	Software Unit Specification
DS	Database Specification
PT	Test Plan
RT	Test Analysis Report
IP	Implementation Procedures

In addition to the standard documents, the PN should include the following useful information:

- List of people involved in the effort: members of the user, developer, ADP, security, communications, training, and maintenance organizations as well as all people consulted about the project.
- Notes from meetings held: dates, attendees, subjects covered, action items, and critical information related to issues discussed, decisions made and their rationale, project organization decisions, and discussions and evaluation of risk factors.
- Transcripts from knowledge engineering sessions and notes on the representation strategies considered and accepted, explaining the mappings from the expert's domain knowledge to the form implemented in the software.
- Descriptions of briefings given: dates, purposes, attendees, copies of material presented, results of briefings.
- Documentation of information collected during the technical tasks (descriptions of the problem, problem environment, written data/knowledge sources, user

community, management attitudes about the problem and ES technology, and all identified risks in these areas).

- Documentation of the risk analysis, prioritization of risks and recommendations for risk resolution; feasibility analysis and recommendations; notes or references to similar ES products or development efforts; and the rough cost estimate.
- Project reviews: dates, attendees, copies of materials presented, result of review including all action items and dates for follow-up (in particular the milestone reviews).
- "Problem" reports and recommendations as a result of the prototype development and T&E.

The PN should be structured to facilitate broad access to its contents by the team members without compromising its more sensitive components (e.g., contractor reviews or internal product evaluations). In the best of all possible situations, the PN should be on-line (perhaps utilizing hypermedia techniques) to support multiple views to meet and manage the varying needs of the project team. Lacking the facilities to maintain the PN on-line, the project manager should keep it in a well-organized binder with plenty of room for insertion and expansion.

B.9 SOFTWARE UNIT SPECIFICATION (US)

The US is a technical document that describes in detail the program or program unit design and implementation.

B.10 SYSTEM/SUBSYSTEM SPECIFICATION (SS)

The SS may be used to describe small systems or large system developments that can be broken down readily into subsystems, each with a separate area of responsibility. This document should be as detailed as possible concerning the environment and design considerations and should present more detailed information than the FD. In particular, this document should define the system/subsystem interfaces.

B.11 TEST ANALYSIS REPORT (RT)

The RT describes the status of the ES after testing and provides a presentation of deficiencies for review by staff and management personnel. Included here should be all additional functionality requests from "problem" reports and T&E activities documented in the Progress Notebook. The RT should be prepared in nontechnical language. The rationale for adding functionality should be clearly stated.

B.12 TEST PLAN (PT)

The PT is a tool for directing the ES testing and contains the orderly schedule of events and list of materials necessary to effect a comprehensive test of the ES. The set of test cases and criteria used in T&E shall be included in the list of materials. Those parts of the PT directed toward the user staff personnel shall be presented in nontechnical language, and those parts directed toward the operations personnel shall be presented in suitable terminology.

B.13 USERS MANUAL (UM)

The UM presents general information about the ES and is directed toward the user organization's general management and staff personnel who have no need for detailed technical information concerning ES implementation or operation. It corresponds roughly to sections 1 and 2 of the UM described in DoD Standard 7935.1.

BIBLIOGRAPHY

[Bardawil 1987] C. Bardawil, L. Fry, S. King, L. Leszcynski, and G. O'Neil, "Artificial Intelligence Software Acquisition Program, Volume I," Prepared by Software Systems Engineering Directorate, Sanders Associates, Inc., A Lockheed Company, Sunnyvale, Ca., RADC Contract No. F30602-85-C-0254, August 1987.

[Bobrow 1983] D. G. Bobrow and M. Stefik, *The LOOPS Manual,* Xerox Corporation, Palo Alto, Ca., December 1983.

[Boehm 1975] B. W. Boehm, C. E. Holmes, G. R. Katkus, J. P. Romanos, R. C. McHenry, and E. K. Gordon, "Structured Programming: A Quantitative Assessment," *Computer*, June 1975, pp. 38–54.

[Boehm 1976] B. W. Boehm, "Software Engineering," *IEEE Transactions on Computers*, December 1976, pp. 1226–1241.

[Boehm 1981] B. W. Boehm, *Software Engineering Economics,* Prentice-Hall, Inc., Englewood Cliffs, N. J., 1981.

[Boehm 1988] Barry W. Boehm, "A Spiral Model of Software Development and Enhancement," *Computer*, Vol. 21, No. 5, May 1988, pp. 61–72.

[Brachman 1978] R. J. Brachman, "A Structural Paradigm for Representing Knowledge," BBN Report No. 3605, Bolt, Beranek & Newman, Cambridge, Mass., 1978.

[Brachman 1985] R. J. Brachman and H. J. Levesque, *Readings in Knowledge Representation,* Morgan Kaufmann Publishers, Inc., 1985.

[Brooks 1987] Frederick P. Brooks, Jr., "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer*, Vol. 20, No. 4, April 1987, pp. 10–19.

[Clancey 1983] W. J. Clancey, "The Epistemology of a Rule-Based Expert System — A Framework for Explanation," *Artificial Intelligence*, 20, 1983, pp. 213–251.

[Clayton 1985] B. D. Clayton, "ART™ Programming Primer," Inference Corporation, Los Angeles, Ca., 1985.

[Clocksin 1981] W. F. Clocksin and C. S. Mellish, "Programming in Prolog," Springer-Verlag, 1981.

[DoD Directive 5000.29] "Management of Computer Resources in Major Defense Systems," DoD, 26 April 1976.

[DoD Directive 5000.29 Draft] "Management of Computer Resources in Major Defense Systems (DRAFT)," DoD, 15 January 1986.

[DoD Directive 7920.1] "Life Cycle Management of Automated Information Systems (AIS)," DoD, 17 October 1978.

[DoD Directive 7920.1 Draft] "Life Cycle Management of Automated Information Systems (AIS)," DoD, 20 June 1988.

[DoD Instruction 7920.2] "Major Automated Information Systems (AIS) Approval Process," DoD, 20 October 1978.

[DoD Standard 2167] "Defense System Software Development," DoD, 4 June 1985.

[DoD Standard 2167A] "Defense System Software Development," DoD, 1 April 1987.

[DoD Standard 7935.1] "Automated Data Systems (ADS) Documentation Standards," DoD, 24 April 1984.

[DoD Standard 7935.1 Draft] "Automated Data Systems (ADS) Documentation Standards (DRAFT)," DoD, 15 January 1988.

[Forgy 1981] C. L. Forgy, "The OPS5 User's Manual," CMU-CS-81-135, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pa., 1981.

[Goldberg 1983] A. Goldberg and D. Robson, "SMALLTALK-80: The Language and Its Implementation," Addison-Wesley, 1983.

[Gorry 1973] G. A. Gorry, "Computer-Assisted Clinical Decision Making," *Methods of Information in Medicine*, 12, 1973, pp. 45–51.

[Hardy 1988] Steven Hardy, "Copernicus™: A Model Knowledge Engineering Tool for Main Stream Environments," Teknowledge, Inc., Palo Alto, Ca., February 1988.

[Hendler 1988] J. Hendler and C. Lewis, "Introduction; Designing Interfaces for Expert Systems," in *Expert System: The User Interface*, Ablex Publications, 1988.

[Hilton 1987] M. L. Hilton, "ERIC: An Object-Oriented Simulation Language," RADC-TR-87-103, Rome Air Development Center, Griffiss Air Force Base, NY, 1987.

[Hollan 1984] J. D. Hollan, E. L. Hutchins, and L. Weitzman, "STEAMER: An Interactive Inspectable Simulation-Based Training System," *The AI Magazine*, Vol. 5, No. 2, 1984.

[Kameny 1989] Iris Kameny, Umar Khan, Jody Paul, and David Taylor, *Guide for the Management of Expert Systems Development: Additional Appendixes*, N-2970-P&L, The RAND Corporation, August 1989.

[Kolodziej 1988] Stan Kolodziej, "The Fate of 4GLs," *ComputerWorld*, February 3, 1988, pp. 25–28.

[Kunz 1984] J. C. Kunz, T. P. Kehler, and M. D. Williams, "Applications Development Using a Hybrid AI Development System," *The AI Magazine*, Vol. 5, No. 3, Fall 1984.

[Lehnert 1978] W. G. Lehnert, "The Process of Question Answering," Lawrence Erlbaum Associates, 1978.

[LoPiccolo 1988] P. J. LoPiccolo, "Expert-System Shells," *Engineering Tools*, May 1988, pp. 64–71.

[Maier 1986] D. Maier, J. Stein, A. Otis, and A. Purdy, "Development of an Object-Oriented DBMS," *Proceedings of OOPSLA '86: Object-Oriented Programming Systems, Languages, and Applications*, The Association for Computing Machinery, September 29–October 2, 1986.

[MAISRC 1987] "Major Automated Information System Review Council (MAISRC), Guidelines for Program Managers," DoD/OSD, June 1987.

[McArthur 1984] D. McArthur, P. Klahr, and S. Narain, *ROSS: An Object-oriented Language for Constructing Simulations*, R-3160-AF, The RAND Corporation, December 1984.

[McKeown 1982] K. R. McKeown, "Generating Natural Language Text in Response to Questions About Database Structure," Ph.D. Dissertation, The Moore School of Electrical Engineering, University of Pennsylvania, 1982.

[Michie 1984] D. Michie, S. Muggleton, C. Riese, and S. Zubrick, "RULEMASTER: A Second-Generation Knowledge-Engineering Facility," *Proceedings of the First Conference on Artificial Intelligence Applications*, IEEE Computer Society, December 1984.

[Morgenstern 1987] M. Morgenstern, "Security and Inference in Multi-level Database and Knowledge-Base Systems," *Proceedings of Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference*, 1987, pp. 357–373.

[Moser 1983] M. G. Moser, "An Overview of NIKL, the New Implementation of KL-ONE," in *Research in Natural Language Understanding, BBN Technical Report 5421*, Bolt, Beranek, and Newman, Inc., Cambridge, Mass., 1983.

[Neches 1985] R. Neches, W. R. Swartout, and J. Moore, "Explainable (and Maintainable) Expert Systems," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985, pp. 382–389.

[Neuron Data 1987] *NEXPERT Object Reference Manual*, Neuron Data Inc., Palo Alto, Ca., 1987.

[Parker 1987] D. Parker, "KEEconnection™: A Bridge Between Databases and Knowledge Bases," Intellicorp, Inc. Mountain View, Ca., 1987.

[Parsaye 1985] W. Parsaye, "The Next 700 Expert System Languages," *COMPCON Sprint 1985, Thirtieth IEEE Computer Society International Conference, Technological Leverage: A Competitive Necessity*, 1985, pp. 108–112.

[Paul 1988] J. Paul, "Self-Revealing Software: A Method for Producing Understandable Systems," Ph.D. Dissertation, University of California at Los Angeles, 1988.

[Prerau 1985], David S. Prerau, "Selection of an Appropriate Domain for an Expert System," *The AI Magazine*, Summer 1985, pp. 26–30.

[Report 1987] "Report of the Defense Science Board Task Force on Military Software," Office of Under Secretary of Defense for Acquisition, Washington, D.C., September 1987.

[Rothenberg 1987] Jeff Rothenberg, Jody Paul, Iris Kameny, James R. Kipps, and Marcy Swenson, *Evaluating Expert System Tools: A Framework and Methodology*, R-3542-DARPA, The RAND Corporation, July 1987.

[Rothenberg 1987] Jeff Rothenberg, Jody Paul, Iris Kameny, James R. Kipps, and Marcy Swenson, *Evaluating Expert System Tools: A Framework and Methodology—Workshops*, N-2603-DARPA, The RAND Corporation, July 1987.

[Royce 1970] W. W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques," *Proceedings WESCON*, August 1970.

[Schoen 1987] S. Schoen and W. Sykes, *Putting Artificial Intelligence to Work—Evaluating & Implementing Business Applications*, John Wiley & Sons, Inc., 1987.

[Shortliffe 1973] E. H. Shortliffe, S. G. Axline, B. G. Buchanan, T. C. Merigan, and S. N. Cohen, "An Artificial Intelligence Program to Advise Physicians Regarding Antimicrobial Therapy," *Computers and Biomedical Research*, 6, 1973, pp. 544–560.

[Shortliffe 1976] E. H. Shortliffe, *Computer-Based Medical Consultations: MYCIN*, American Elsevier, 1976.

[Shortliffe 1984] E. H. Shortliffe, "Explanation Capabilities for Medical Consultation Systems," *Proceedings of AAMSI Congress '84*, San Francisco, May 1984, pp. 193–197.

[Scott 1977] A. C. Scott, W. J. Clancey, R. Davis, and E. H. Shortliffe, "Explanation Capabilities of Production-Based Consultation Systems," *American Journal of Computational Linguistics*, 1977, pp. 338–362.

[Swartout 1981] W. D. Swartout, "Producing Explanations and Justifications of Expert Consulting Programs," Ph.D. Dissertation, Massachusetts Institute of Technology, January 1981.

[Szolovitz 1987] P. Szolovitz, "Expert Systems Tools and Techniques: Past, Present, and Future," in *AI in the 1980s and Beyond: an MIT Survey*, MIT Press, 1987.

[Tracz 1988] Will Tracz, "Confessions of a Used-Program Salesman—Programming-In-The-New," *Computer*, Vol. 21, No. 5, May 1988, p. 74.

[Waterman 1986] D. A. Waterman, *A Guide to Expert Systems,* Addison-Wesley Publishing Company, 1986.

[Winston 1987] P. H. Winston, "The Commercial Debut of Artificial Intelligence," in *Applications of Expert Systems,* ed. J. Ross Quinlan, Addison-Wesley Publication Company, 1987.

[Wright 1983] J. M. Wright and M. S. Fox, *SRL/1.5 User Manual,* Robotics Institute, Carnegie-Mellon University, Pittsburgh, Pa., December 1983.

[Young 1987] P. A. Young, "Defense Logistics Standard Systems Functional Requirements," Report DL502R1, Logistics Management Institute, Bethesda, Md., March 1987.